International Journal of Innovative Research and Scientific Studies, 8(3) 2025, pages: 1205-1221

IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

Optimizing machine learning models for tractor surface recognition: Performance and model size reduction for embedded systems

Phummarin Thavitchasri[1], Dechrit Maneetham[1*], Padma Nyoman Crisnapati[1]

[1]*Department of Mechatronics Engineering, Rajamangala University of Technology Thanyaburi, Pathum Thani 12110, Thailand.*

Corresponding author: Dechrit Maneetham (*Email: dechrit_m@rmutt.ac.th*)

Abstract

This research applied surface recognition in autonomous tractors is essential for efficient navigation across diverse terrains of asphalt, gravel, and soil by using machine learning models and processing data acquisition from the BNO055 IMU sensor. In addition, this aims to evaluate and optimize various machine learning models, including Logistic Regression, K-Nearest Neighbors (KNN), SVC, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, and XGBoost, for surface classification, with a focus on reducing model size without sacrificing classification accuracy. The research applies model pruning techniques to optimize these models for TinyML environments. The results demonstrate that XGBoost and Random Forest achieve high classification accuracy but have large model sizes, which can be mitigated through pruning, significantly reducing their size while maintaining performance. This study provides valuable insights into the trade-offs between model size and accuracy, contributing to the development of more efficient models for embedded systems in autonomous tractors. The findings highlight the potential of model pruning in enabling real-time surface recognition in resource-constrained environments, offering a scalable solution for deployment in agricultural machinery. The best results demonstrate 91 percent accuracy for XGBoost; Random Forest, on the other hand, has a very large trimmed model size (19,603 KB) and more efficient operations than other machine learning models for surface classification.

Keywords: Autonomous tractors, Embedded systems, Internet of things, Machine learning, Model pruning, Surface classification, TinyML.

DOI: 10.53894/ijirss.v8i3.6770
Funding: This study received no specific financial support.
History: Received: 10 March 2025 / **Revised:** 14 April 2025 / **Accepted:** 17 April 2025 / **Published:** 06 May 2025
Copyright: © 2025 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Transparency: The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.
Acknowledgment: The author would like to express sincere gratitude to Rajamangala University of Technology Thanyaburi (RMUTT) for their continuous support and guidance throughout this research. Special thanks are extended to the faculty members, research staff, and students of the Mechatronics Engineering International Program at RMUTT for their invaluable contributions, insightful feedback, and encouragement, which played a key role in the successful completion of this study. Additionally, the research would not have been possible without the resources and facilities provided by the university. Their commitment to fostering an innovative research environment has been instrumental in advancing this work.
Publisher: Innovative Research Publishing

1. Introduction

Because it enables vehicles, such tractors, to navigate a range of terrains, including soil, gravel, and asphalt, surface identification is crucial to autonomous systems. Surface recognition technologies enable autonomous tractors to adjust their movement and control strategies in real-time based on the terrain they are traversing [1]. The ability to distinguish between different surfaces can improve overall performance, safety, and operational effectiveness, which is especially important for agricultural equipment [2]. However, there is a challenge with implementing these systems on embedded devices, including tractors with Internet of Things (IoT) capabilities [3]. These systems must manage the limitations of embedded devices, such as constrained memory and compute power, while handling real-time surface classification utilizing sensor data from accelerometers, gyroscopes, and magnetometers [4, 5]. Hence, achieving high classification accuracy while optimizing models for low-resource environments is a critical problem that needs to be addressed [6].

Many studies have been conducted on the use of machine learning (ML) for surface detection and terrain categorization in autonomous vehicles [1]. For this job, a variety of models have been employed, including Random Forest, XGBoost, and K-Nearest Neighbors (KNN) [7], all of which have demonstrated exceptional surface classification accuracy. For instance, Kabir et al. [8] employed Random Forest to accurately categorize topography in autonomous vehicles, but they neglected to focus on model size or optimization for deployment on low-resource devices. Similar to this, Cumbajin et al. [9] classified surfaces using CNN, focusing on accuracy and neglecting the issue of reducing the model size for embedded system deployment. These models are well known for their ability to forecast outcomes, but because of their size, they are not as well suited for real-time implementation in embedded systems, such as IoT-enabled tractors, where memory and processing capacity are frequently constrained [10].

However, several studies, such as Njor et al. [11], have focused on model pruning techniques to reduce the size of machine learning models like Random Forest and XGBoost for TinyML applications. The impact of pruning techniques on tractor surface classification performance is yet unknown, despite the fact that pruning has shown promise in reducing model size [12]. The gap in the research is the understanding of how to balance model size reduction with classification accuracy, particularly when deploying on embedded devices with stringent resource constraints. This gap is an excellent opportunity to carry out more research on the optimization of machine learning models for real-world applications in autonomous tractors.

The primary objective of this work is to evaluate several machine learning models, such as Logistic Regression [13] K-Nearest Neighbors (KNN) [14] SVC [15] Decision Tree [16] Random Forest [17] Gradient Boosting [18] AdaBoost [19] and XGBoost [20] for the task of surface recognition in autonomous tractors. The primary objective is to optimize these models for deployment in TinyML contexts of embedded systems. In order for the model to perform well on Internet of Things-based tractor systems, this study specifically identifies the model that provides the optimal balance between low model size and high classification accuracy.

To achieve that objective, this study examines and assesses a range of machine learning models, including more traditional models like Logistic Regression and more complex algorithms like XGBoost. The models will be trained using accelerometer, gyroscope, and magnetometer sensor data, and their performance will be evaluated using classification accuracy, precision, recall, and F1 scores [21]. Model pruning strategies will also be employed to reduce the size of the models in order to assess the effects on accuracy and model size [22]. The pruned models' and their unpruned counterparts' performances will be contrasted.

According to early findings from training and assessing the models, XGBoost and Random Forest exhibit the highest classification accuracy, attaining high precision and recall levels. These models do, however, also have larger model sizes, which makes implementation on embedded devices more difficult. The model size is greatly reduced after using pruning strategies, especially for XGBoost, without a discernible decline in classification performance. This illustrates how pruning strategies can be used to optimize models for TinyML applications.

The findings show that although Random Forest and XGBoost provide the best classification accuracy, their large model sizes are a drawback for the deployment of embedded systems. These models' size was successfully decreased while retaining excellent accuracy through the use of pruning procedures, which qualified them for use on IoT-enabled tractors. This study emphasizes how crucial it is to optimize machine learning models for real-time, resource-constrained contexts by taking into account both accuracy and model size. To further increase the effectiveness of these models without sacrificing their performance, future research could investigate more sophisticated pruning techniques and model quantization.

2. Materials and Methods

The materials and methods of research will follow overview process as Figure 1.

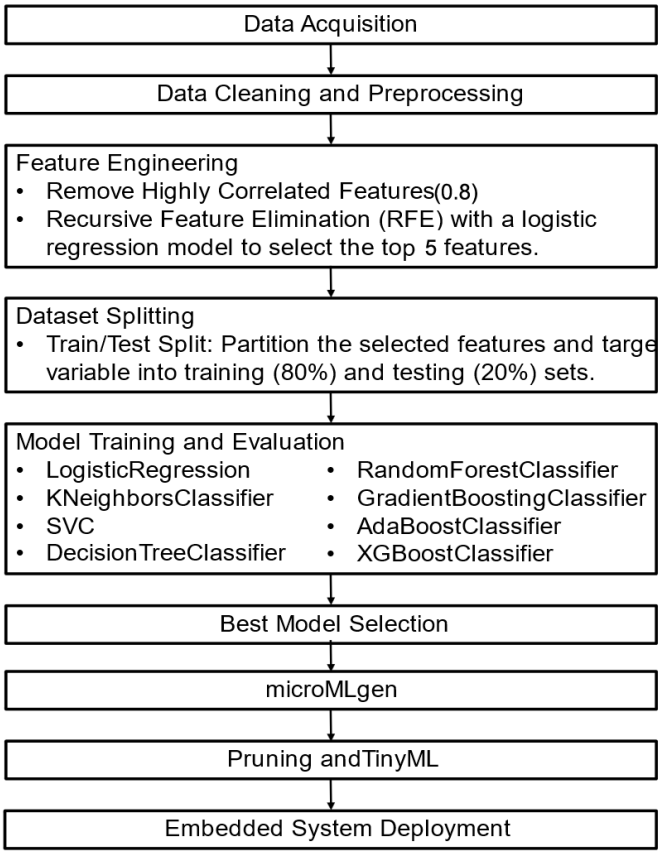

Figure 1.
Research Overview.

2.1. Data Acquisition Cleaning, and Preprocessing

The dataset used in this study includes data from a BNO055 IMU sensor at a tractor robot model in Figure 2, the overall parameter capturing accelerometer, gyroscope, and magnetometer readings under different totals surface 7 events as Table 1.

Table 1.
Performance Evaluation Results.

Output data Collections	Type of event	Type for track and surface
1	Event 1	Around on asphalt surface
2	Event 2	Around on gravel surface
3	Event 3	Zigzag on gravel surface
4	Event 4	Around on concrete and gravel surface
5	Event 5	Zigzag on concrete and gravel surface
6	Event 6	Around on Soil + Grass surface
7	Event 7	Zigzag on Soil + Grass surface

The dataset consists of the following features:

- eu-X, eu-Y, eu-Z: Euler angles
- acc-X, acc-Y, acc-Z: Accelerometer data
- gyro-X, gyro-Y, gyro-Z: Gyroscope data
- mag-X, mag-Y, mag-Z: Magnetometer data
- output: Class labels representing different surface types

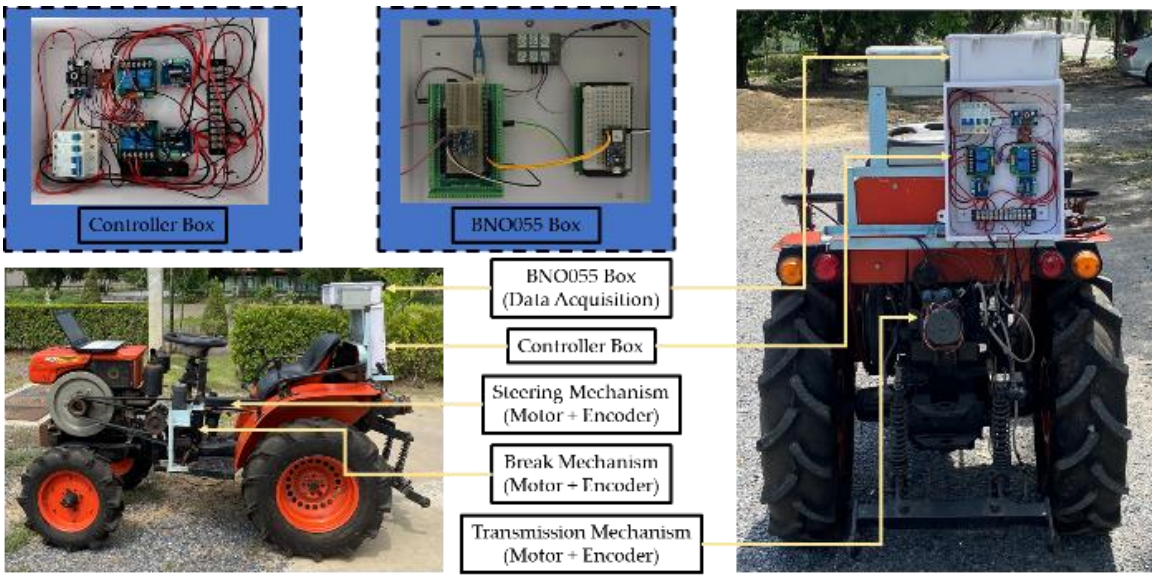

Figure 2.
Tractor Robot Model.

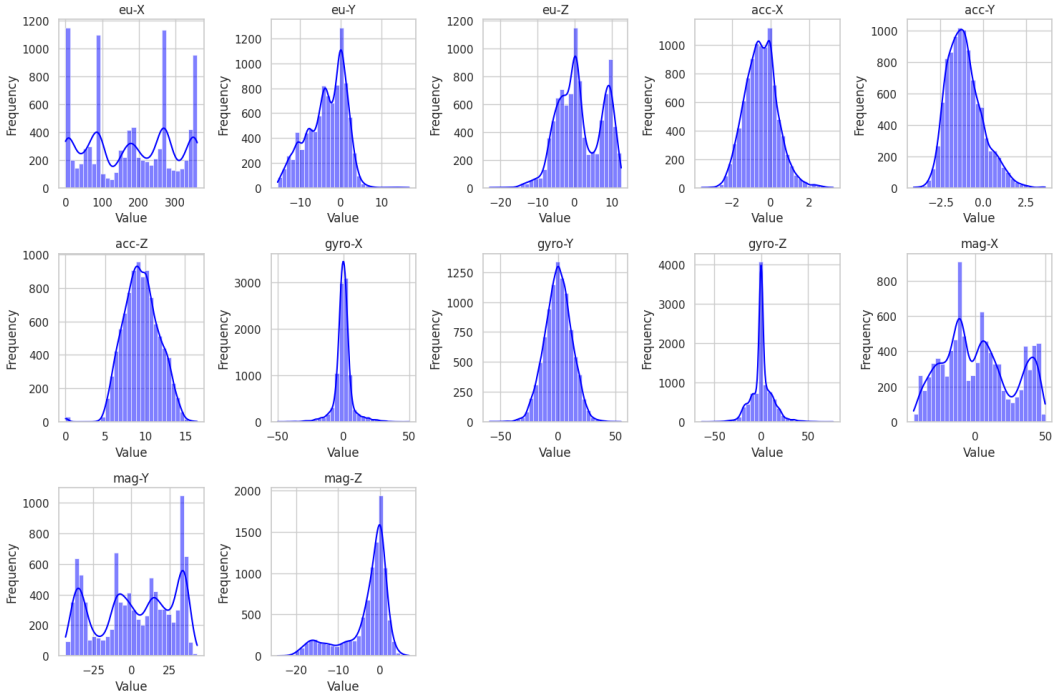

Figure 3.
Data Visualization.

Figure 3, shown histograms and kernel density estimations (KDE) for several features in the dataset, particularly the IMU sensor readings, are shown in Figure 3. The sensor's orientation in three dimensions is represented by the Euler angle characteristics (eu-X, eu-Y, and eu-Z). The eu-X and eu-Z distributions seem to be very uniform, indicating that the sensor data is dispersed among several orientations. The skewed distribution of eu-Y, on the other hand, suggests that the sensor data are concentrated in particular orientations. This could reveal details on how the tractor moved or the setting where the data was gathered.

With peaks close to zero and a broad spread on the positive side, the accelerometer data (acc-X, acc-Y, and acc-Z) exhibit skewed distributions, especially for acc-X and acc-Y. This implies that specific acceleration ranges are where the tractor's motion, or even outside forces operating on it, are focused. A significant percentage of the acc-X, acc-Y, and acc-Z data points fall near zero, which may signify times during data collection when there is relative stability, such as when the tractor is not moving or the IMU sensor is comparatively stationary.

Angular velocity is measured using the gyroscope data (gyro-X, gyro-Y, and gyro-Z). There was little rotation during the data collection period, as indicated by the large concentrations at zero in the distributions for these features. Sharp spikes, especially for gyro-Z, are present in some of the distributions, though, and may be a sign of sensor irregularities or abrupt rotational motions. To lessen the impact of these spikes on subsequent machine learning models, further research or preprocessing (such as outlier detection) may be necessary.

There is a lot of variability throughout the range and some concentration close to zero in the magnetometer source. This can be a sign of noise in the sensor data or of other elements, such as metal items or electrical interference that alter the magnetic field values. Prior to training machine learning models, further preprocessing (such as filtering) may be necessary to eliminate noise from the data, much like with the other sensor readings.

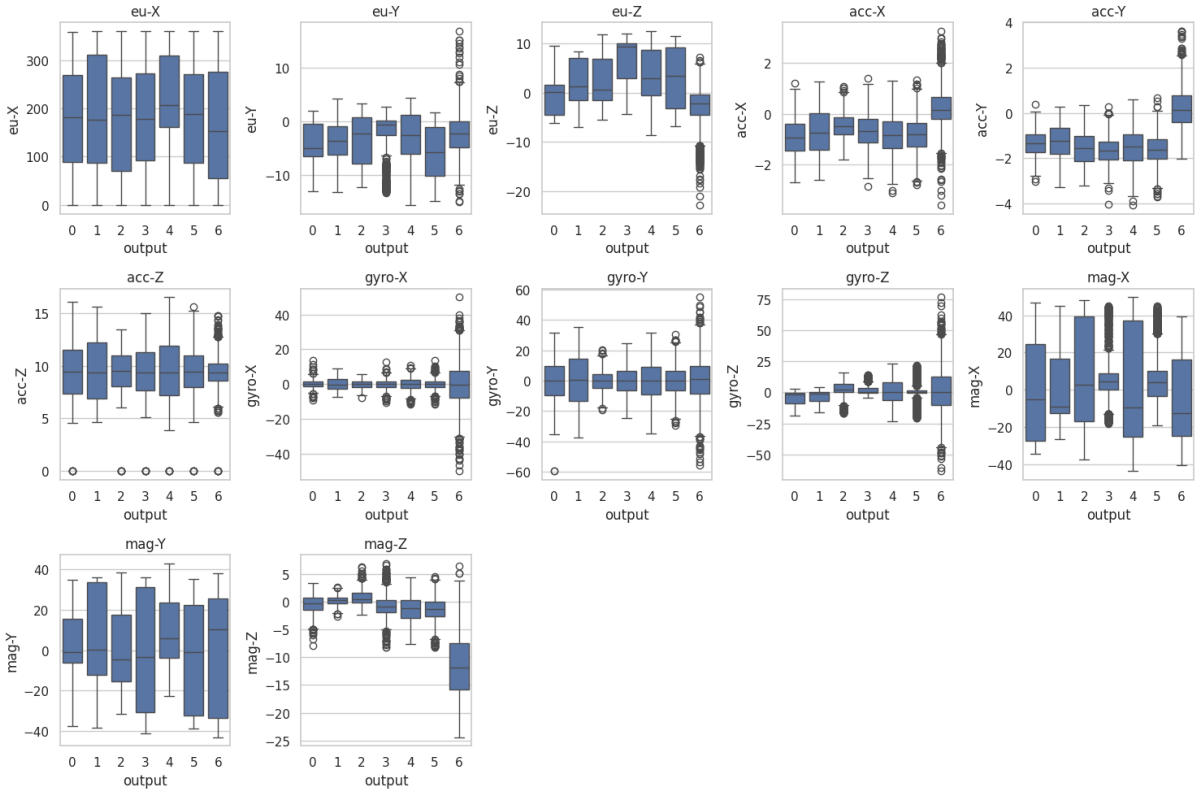

Figure 4.
Boxplot Data Visualization.

Each feature in the dataset is represented by a boxplot in Figure 4, which is grouped by the output variable and corresponds to various surface types (0–6). The values for a particular characteristic are distributed among the several surface types in each boxplot. Data distribution can be visually summarized with boxplots, which show the median, range, and outliers.

The orientation of the sensor in three dimensions is represented by the Euler angle characteristics (eu-X, eu-Y, and eu-Z). There are outliers in both the eu-X and eu-Z boxplots, which demonstrate notable diversity among the various surface types. These discrepancies imply that the orientation of the sensor varies more significantly on some surfaces than others. In the Y-axis orientation, eu-Y has a more concentrated distribution, suggesting more reliable sensor readings. This could be a reflection of the surfaces' characteristics, as some lead to more dynamic shifts in the orientation of the sensor.

There is greater fluctuation in some surface types, particularly in acc-X and acc-Y, according to the accelerometer data. For surface type 0, for instance, acc-Z has a rather tight distribution, indicating that the Z-axis acceleration varies less on this surface. The acc-X and acc-Y axes, on the other hand, exhibit broader dispersion across various surfaces, suggesting that the tractor's movement dynamics vary more depending on the kind of surface.

High variability is seen in the gyroscope features (gyro-X, gyro-Y, and gyro-Z), especially in gyro-Z. Outliers for surface types 1 and 5 are visible in the gyro-Z boxplot, suggesting significant rotational movements or abrupt direction changes. This indicates that the tractor rotates more on these surfaces as a result of the nature or roughness of the surface. Conversely, gyro-X and gyro-Y exhibit more concentrated distributions, indicating less noticeable rotational movements along the X and Y axes.

There is significant variability and outliers for surface types 1 and 5 in the magnetometer data (mag-X, mag-Y, and mag-Z), especially for mag-X and mag-Y. The presence of different terrain characteristics that affect the magnetometer readings could be the cause of the tractor experiencing fluctuating magnetic field conditions. The variation between surfaces suggests that the magnetometer may be susceptible to variations in the surrounding environment, such as the makeup of the terrain or adjacent objects.

The sensor readings' variations across various surfaces are displayed in the box plots. Certain characteristics, including gyro-Z and acc-Y, exhibit more noticeable fluctuation between surface types, suggesting that they might be more effective at differentiating between surfaces. Features like acc-Z and gyro-X, on the other hand, seem to be more consistent across surfaces, which may indicate that they are less useful for classification. Outliers, especially in the gyroscope and magnetometer data, suggest that there may be intermittent sensor events or sensor noise, which may need to be fixed during preprocessing in order to increase model accuracy.

The selection of the most useful features for machine learning tasks will be aided by these insights about the distribution of features across surface types. To increase the classification model's efficacy, the following actions might include addressing outliers, normalizing these features, and utilizing feature selection strategies. The fact that different surfaces have different sensor readings also raises the possibility that some surfaces have unique sensor characteristics, which might be exploited to improve surface recognition.

The associations between each pair of attributes in the dataset are displayed in Figure 5, which is a matrix of scatter plots and histograms. The distribution of every single feature along the diagonal is also included. Understanding the feature distribution and seeing any relationships between them, as well as recognizing the patterns for each output class, which corresponds to various surface types in your dataset are two areas where this kind of visualization is quite helpful.

All of the features are all included in the pair plot. While the histograms along the diagonal display the individual distributions of each characteristic, the scatter plots shed light on the pairwise correlations between each pair of features. We can see from the visualizations how the features relate to one another in the various output classes, which correspond to various surface kinds.

The pairwise scatter plots show a number of intriguing connections. For example, some feature pairs, such as acc-X and acc-Y, exhibit discernible clusters or patterns, suggesting a possible linear relationship between the two features. Some scatter plots, on the other hand, lack a discernible pattern, indicating little to no association between those feature pairs. This can aid in feature selection because highly correlated feature pairs may be redundant and can be removed to simplify the model.

Additionally, it is simple to compare how features behave on various surface types due to the plot's usage of different colors to denote the output class. Certain features, such as gyro-X and acc-Z, exhibit distinct variations in distribution for various surface types, which may indicate that they are more representative of particular surfaces. However, characteristics like mag-Y and mag-Z exhibit greater overlap between surface types, suggesting that they might not be as useful in differentiating between various surfaces.

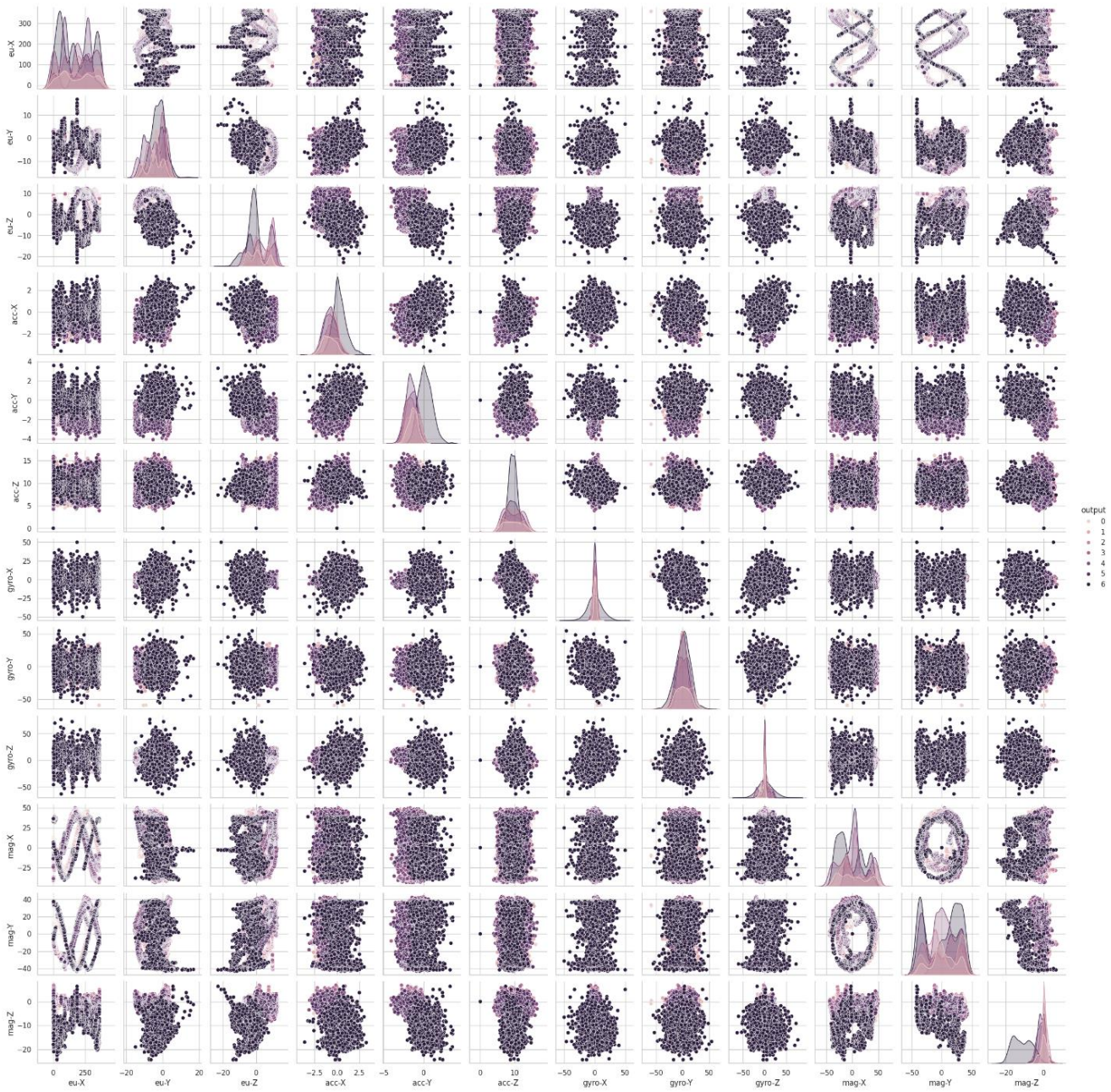

Figure 5.
Scatter Plot Data Visualizations.

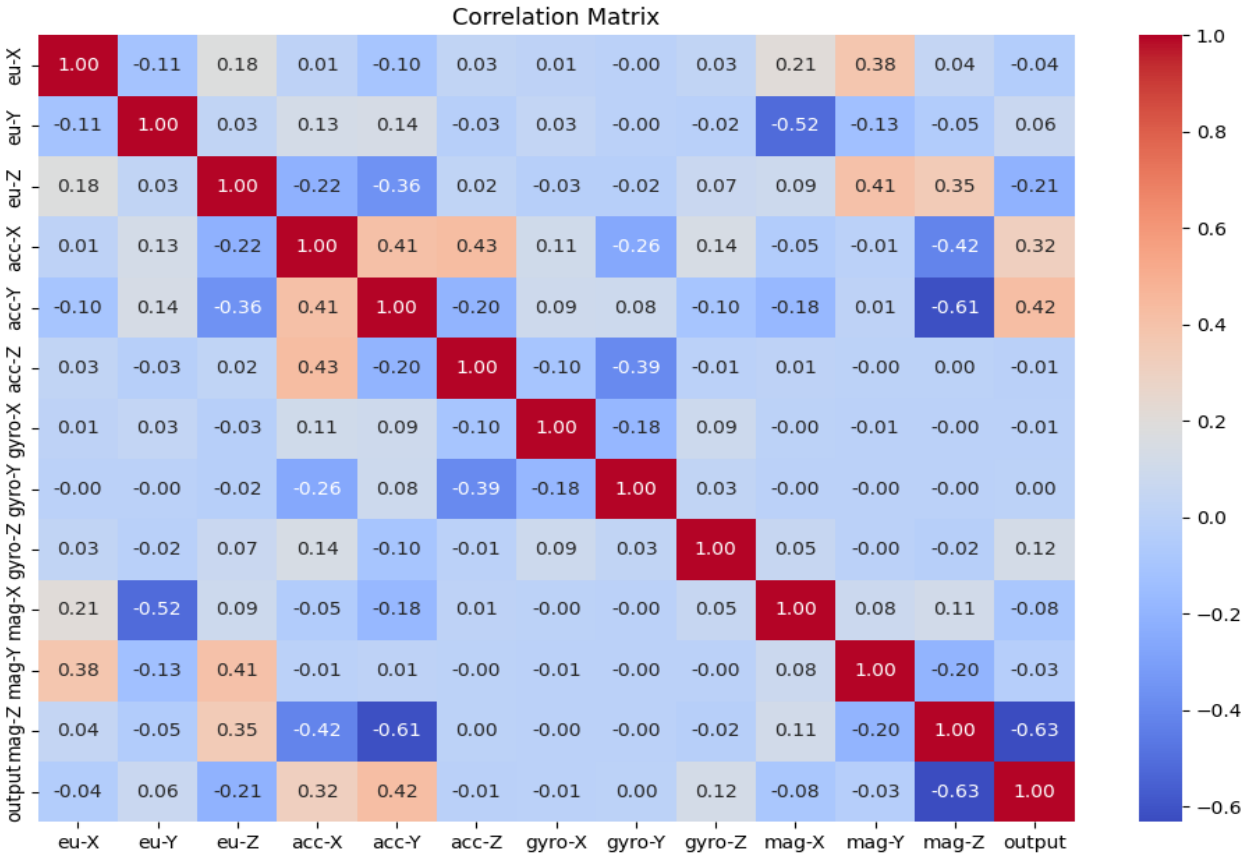

Figure 6.
Correlation Matrix.

A correlation matrix, shown in Figure 6. Features like acc-Z and gyro-Z, for instance, show a somewhat positive correlation of 0.41, indicating that variations in angular velocity along the Z-axis are somewhat correlated with variations in acceleration along the Z-axis. A similar pattern in the rotational movements along the X and Y axes may be implied by the moderately positive correlation of 0.39 between gyro-X and gyro-Y. However, characteristics like acc-Z and acc-Y show a high negative correlation of -0.61, indicating that acceleration along the Y-axis tends to decrease as acceleration along the Z-axis increases. This could be because of opposing forces or differential movements.

Additionally, other feature pairs, like gyro-X and mag-X and gyro-Y and mag-Y, have weak or no correlations with one another, indicating that they may represent independent information and may not significantly affect one another. Apart from acc-Z and mag-Z, the output feature, which most likely represents various surface types, exhibits modest correlations with most sensor features.

As a result of these realizations, highly connected traits must be eliminated. Multicollinearity may result from features like acc-Z and acc-Y, which have a large negative correlation, or other pairings with high correlation values, such as gyro-X and gyro-Y. When characteristics are highly correlated, multicollinearity occurs, which can make regression models unstable and less interpretable. The model's efficiency is increased, and redundancy is decreased by eliminating strongly correlated features, which simplifies the model, reducing the likelihood of overfitting and enhancing its capacity for generalization.

Feature selection by Recursive Feature Elimination (RFE) is essential. Recursively eliminating the least significant features depending on model performance is how RFE chooses the most crucial ones. By keeping only the most pertinent features for prediction, this procedure helps to reduce the dimensionality of the dataset, which can increase the accuracy of the model and lessen overfitting. By using RFE, we ensure the model only considers the most important features, increasing its predictive ability and computational efficiency.

2.2. Feature Engineering

To lessen multicollinearity, highly correlated features (threshold > 0.8) were eliminated using feature engineering. Additionally, the top five most significant features were chosen using a logistic regression model and Recursive Feature Elimination (RFE). By reducing the dataset's dimensionality, this step enhanced the generalization and training efficiency of the model. Data cleaning and preprocessing, with special emphasis on feature engineering, are the main goals of the methodology depicted in the graphic. Recursive Feature Elimination (RFE) using a logistic regression model to choose the top 5 features and eliminating highly correlated features (threshold = 0.8) are the two primary procedures outlined in this methodology.

2.2.1. Remove Highly Correlated Features (Threshold = 0.8)

Features (or variables) in any dataset may occasionally have a strong correlation with one another. This indicates that modifications to one feature are mirrored in modifications to another. Multicollinearity problems in machine learning models can be brought on by high correlations, particularly those that are above a particular threshold (in this example, 0.8) [23]. High levels of correlation across independent variables can lead to multicollinearity, which makes it challenging to identify the precise impact of each characteristic on the target variable. As a result, the model may overfit to the data and produce unstable model coefficients, which would reduce its generalizability [22].

Our goal is to eliminate duplication by eliminating features with a high correlation (above 0.8). Removing one of the highly associated features reduces the feature space without significantly reducing the amount of information provided. As a result, models become more stable and interpretable by lowering the likelihood of overfitting and multicollinearity. The linear link between two properties is commonly measured using the Pearson correlation coefficient. If the absolute value of r_{xy} exceeds 0.8, it suggests that features X and Y are highly correlated and one of them may be removed. It is computed as Equation 1, where r_{xy} is the correlation matrix coefficient between features *X* and *Y*. X_i and Y_i are individual sample values for features *X* and *Y*. $\bar{X}$ and $\bar{Y}$ are the means of *X* and *Y*.

$$r_{xy} = \frac{\sum(X_i - \bar{X})(Y_i - \bar{\bar{Y}})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \quad (1)$$

2.2.2. Recursive Feature Elimination (RFE) with Logistic Regression to Select the Top 5 Features

A feature selection method called Recursive Feature Elimination (RFE) constructs a model to assess performance at each stage while recursively eliminating the dataset's least significant features. Finding and keeping the most important properties for the predictive model is the aim. In RFE, a model is fitted, and features are ranked according to their significance [24]. Until the required number of features is kept, the least significant characteristics are eliminated, and the procedure is repeated. A linear model for binary classification, logistic regression can also be employed for multi-class classification. Each feature is given a weight, and the feature's significance is indicated by the weight's magnitude. Features with lower weights are seen as less significant. First, match the dataset to the logistic regression model. Sort features according to their absolute importance, or coefficients. Refit the model using the remaining characteristics after removing the least significant one. Continue steps two and three until the top five features are chosen. The feature selection logistic regression model can be expressed as Equation 2, where $p(y = 1|X)$ is the probability of the target variable being 1 given the input features *X*. β_0 is the intercept, $\beta_1$are the coefficients for each feature X_i, indicating the importance of the respective features *e* is the base of the natural logarithm.

$$p(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i X_i)}} \quad (2)$$

By analyzing the absolute values of the features' coefficients (βi), the Recursive Feature Elimination (RFE) method iteratively eliminates the least significant features according to the values of their associated coefficients. The most significant features are those with the highest absolute coefficients, and these are kept in the model. First, the RFE algorithm ranks every feature in a logistic regression model according to the size of its coefficients. The significance of the remaining features is then recalculated when the least significant feature is eliminated. Until the desired number of characteristics is obtained in this case, choosing the top five features, this process is repeated. The features that make the biggest contribution to the predictive power of the model are the last set to be chosen.

Recursive feature elimination (RFE) and the removal of highly correlated features are essential feature engineering procedures that guarantee a more reliable and effective model. The model may overfit to the data and become harder to understand when features are highly linked. By eliminating these superfluous characteristics, we lower the dataset's dimensionality without sacrificing important information, which improves generalization and speeds up training.

By determining which elements are most crucial for prediction and removing the least relevant ones, RFE, on the other hand, improves the feature selection procedure. This guarantees that the model only employs the most useful variables, increasing accuracy and decreasing overfitting. It is particularly crucial when working with a high number of features. When both approaches are combined, the dataset becomes cleaner and more effective, which improves machine learning model performance.

To sum up, the two feature engineering methods, removing highly correlated features and using RFE, are crucial for producing an ideal model that is both precise and understandable, especially when it comes to machine learning applications for tractor system is surface recognition.

2.3. Model Selection and Evaluation

This segment assessed many machine learning models for surface classification in order to identify the most effective model for identifying distinct surface types from the sensor data. K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Decision Tree, Random Forest, Gradient Boosting, AdaBoost, XGBoost, and Logistic Regression are among the models taken into consideration in this study. The theoretical justifications and formulas for each of these models are provided below.

2.3.1. Logistic Regression

A probability value between 0 and 1 is the output of the linear model known as logistic regression, which is used for binary classification. It calculates the correlation between the probability of the binary outcome (class 0 or class 1) and the

independent features. The model maps any input to a probability by assuming a logistic (sigmoid) function. By employing optimization techniques like gradient descent to minimize the logistic loss function, the coefficients βi are discovered [25].

2.3.2. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a classification technique that is non-parametric. Based on the majority class of its k-nearest neighbors, it determines a data point's class. The "closeness" of the data points in the feature space is determined using the distance metric, which is often Euclidean. The projected class for a specific test point x_{test} is ascertained as Equation 3, where $y_i, y_{i_2}, \dots, y_{i_k}$ are the labels of the *k* nearest neighbors of x_{test}. The distance between the test point x_{test} and each training point x_i is typically computed using Euclidean distance Equation 4, where x_{test}, x_i and $x_{i,j}$ are the feature values of the test point and the i^{th} neighbor, respectively [26].

$$\hat{y} = mode(y_i, y_{i_2}, \dots, y_{i_k}) \quad (3)$$

$$d(x_{test}, x_i) = \sqrt{\sum_{j=1}^{n} \left(x_{test}, j - x_{i,j}\right)^2} \quad (4)$$

2.3.3. Support Vector Classifier (SVC)

A supervised learning approach called the Support Vector Classifier (SVC) locates a hyperplane in a high-dimensional space that maximum divides the classes. The objective is to minimize the classification error and maximize the margin between the classes. Both linear and non-linear kernels (such as the radial basis function kernel) can be utilized with SVC. The SVC model seeks to identify a hyperplane for a linearly separable case (Equation 5), where *w* is the weight vector that is orthogonal to the hyperplane. X is the feature vector; b is the bias term, and $\frac{1}{\|w\|}$ as the margin. The goal is to maximize the margin while ensuring that each data point is classified correctly. The decision rule for classification is given by Equation 6, where the kernel trick is often used to transform the data into a higher-dimensional space to handle non-linear separations, defined by $K(x_i, x_j)$ [27].

$$w^T x + b = 0 \quad (5)$$

$$f(x) = sign(w^T x + b) \quad (6)$$

2.3.4. Decision Tree

A non-linear classifier called a decision tree divides the data into subsets according to the feature values. Each split is created by selecting the trait that best separates classes or yields the most information gain. The choice to divide at each node in the tree is determined by either maximizing information gain or reducing the Gini impurity. For a given S, the Gini impurity is defined as Equation 7, were p_i is the proportion of elements of class I in the set S. The information gain is calculated as the difference between the entropy of the parent node and the weighted sum of the entropy of the child nodes [28].

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2 \quad (7)$$

2.3.5. Random Forest

To increase classification accuracy, Random Forest, an ensemble learning technique, mixes several decision trees. A random subset of the data is used to train each tree, and the outputs of all the trees are combined to create predictions. The Random Forest model aggregates each tree Ti individual predictions to produce a prediction as shown in Equation 8, where $T_i(x)$ is the prediction made by the i^{th} decision tree, and *M* is the total number of trees in the forest [29].

$$\hat{y} = mode(T_1(x), T_2(x), \dots, T_M(x)) \quad (8)$$

2.3.6. Gradient Boosting

Gradient Boosting is an ensemble method that creates trees one after the other, fixing the mistakes of the preceding tree. Each new tree matches the residuals (errors) of the preceding trees, and the model minimizes a loss function using a gradient descent process. A new tree hm(x) that optimizes the loss function is added to the model at each step m Equation 9 where $F_m(x)$ is the model at the m^{th} step, $h_m(x)$ is the new tree added at step *m*, and η is the learning rate. The loss function $L(y, F_m(x))$ is typically the mean squared error for regression or log-loss for classification [30].

$$F_m(x) = F_{m-1}(x) + \eta h_m(x) \quad (9)$$

2.3.7. AdaBoost

Another ensemble method called AdaBoost (Adaptive Boosting) builds a stronger model by combining weak learners, usually decision trees. By changing the weights of data that are mistakenly identified after each iteration, it concentrates more on data points that are challenging to classify. Following each iteration, the AdaBoost algorithm modifies the weight of every data point. A weighted aggregate of all weak learner guesses makes up the final prediction (Equation 10), where $h_m(x)$ is the weak learner at step *m*, and α_m is the weight assigned to $h_m(x)$, based on its performance [31].

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x) \quad (10)$$

2.3.8. XGBoost

An enhanced version of gradient boosting is called XGBoost (Extreme Gradient Boosting). It is intended to be both scalable and extremely efficient. Because XGBoost uses regularization (L_1 and L_2) to avoid overfitting, it works especially well with big datasets. The following objective function is optimized by XGBoost Equation 11, where *l* is the loss function,

$\hat{y}_i$ is the prediction for sample *i*, and $\Omega(f_k)$ is the regularization term to penalize overly complex trees. To improve model generalization, XGBoost uses regularization in conjunction with gradient boosting. Considering the trade-offs between accuracy, model complexity, and computational economy, these machine learning models were chosen for evaluation in order to determine which model performs the best for surface categorization [32].

$$L(\theta) = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (11)$$

2.4. Model Pruning and TinyML Conversion

Model size was a crucial factor for deployment on embedded systems, particularly in resource-constrained contexts like Internet of Things (IoT) devices and other edge devices, in addition to assessing each machine learning model's performance. Because of their frequently constrained memory and processing capabilities, these systems require model optimization to function well without sacrificing accuracy. In order to solve this, we used model pruning techniques to make the models smaller without sacrificing speed, which allowed them to be deployed embedded in TinyML apps.

2.4.1 Model Pruning

Model pruning is a method for shrinking a machine learning model's size and complexity, usually by removing components that don't significantly improve the model's performance. In order to make the model more appropriate for embedded deployment, where computational resources are scarce, it is intended to be made simpler while preserving or increasing its accuracy [33, 34].

Pruning is applicable to different parts of a machine learning model. Pruning might entail lowering the number of trees in the ensemble or restricting the depth of individual trees for decision tree-based models such as Random Forest and XGBoost. Pruning neural networks entails eliminating individual neurons or less significant weights. This results in a smaller model size by lowering the number of parameters in the model.

Pruning decision tree models usually entails limiting the maximum tree depth, the least number of samples needed to divide a node, and the minimal impurity. Reducing the entropy or Gini impurity at each node is frequently the basis for the pruning choice. A node's Gini impurity can be found using Equation 12, where p_i is the proportion of the data points of class *i* in the node *t*, and *C* is the number of classes in the target variable.

$$Gini(t) = 1 - \sum_{i=1}^{C} p_i^2 \quad (12)$$

When pruning, we try to cut off branches that don't substantially increase the Gini impurity or information gain. For instance, the branch may be cut to lessen the complexity of the tree if a split does not result in a significant decrease in impurity. In neural networks, pruning entails eliminating weights according to their significance. A common trimming approach determines the weight's importance based on its magnitude. Weights that are near zero can be trimmed because they are deemed less significant.

2.4.2. TinyML Conversion

The use of machine learning models on tiny, resource-constrained devices, like microcontrollers and Internet of Things gadgets, is known as "tinyML". Traditional machine learning models must be compressed to operate well in these settings because these devices usually have limited processing power, memory, and storage. Two methods of model optimization are used in TinyML. use methods like knowledge distillation, quantization, and pruning to reduce the size of the model. By using specific hardware accelerators (such as TensorFlow Lite for Microcontrollers), the model can run effectively on embedded devices with constrained resources [35].

It is possible to lower the size of the models employed in this study (Random Forest and XGBoost with the best accuracy) by pruning them before deployment. TinyML devices frequently have extremely little memory, therefore the size reduction is especially crucial. For instance, the quantity of trees and depth of each tree may cause an XGBoost model to be quite enormous. Pruning the trees lowers the total amount of memory needed to hold the model as well as the number of nodes. Following pruning, models are usually transformed into an embedded system-optimized format, like TensorFlow Lite for deep learning models or specialized formats for decision trees, like the binary format used by XGBoost.

One method that lessens the accuracy of the model's weights and activations is quantization. Quantization drastically reduces the model's memory footprint by converting weights from 32-bit floating-point numbers to 8-bit integers, which is what a normal model might employ. After that, the quantized model is optimized for embedded hardware execution. The following scaling factor is used in quantization to transform the weights from floating-point to integer values (Equation 13), where *w* is the original floating- point weight. *s* is the scaling factor determined by the range of the weights. The round function ensures that the weight is converted to the nearest integer [36].

$$w_{quantized} = round\left(\frac{w}{s}\right) \quad (13)$$

On embedded devices, this decrease in accuracy enables models to operate more quickly and use less memory. The model is transformed into a format that can be operated effectively on TinyML devices when pruning and quantization are completed. The conversion process entails converting the model into an optimal format and making sure it works with the desired hardware.

2.4.3. Benefits of Pruning and TinyML Conversion

When implementing machine learning models on embedded systems, the combination of TinyML conversion and model pruning offers several advantages. The model's size is further reduced by quantization and pruning, which both help the

model fit into the limited memory of TinyML devices. Faster inference speeds from smaller models are essential for real-time applications in settings with limited resources. Smaller models use less processing power, which is crucial for battery-powered gadgets. By eliminating components of the model that are unlikely to significantly contribute to the prediction, pruning can also aid in lowering overfitting. In conclusion, model pruning makes machine learning models more suited for deployment on devices with limited resources by reducing their size and complexity. These models are then optimized for effective embedded system execution using TinyML conversion. Without compromising speed, these methods are crucial for enabling real-time machine learning applications in embedded systems and the Internet of Things, such as surface identification in driverless cars [37].

3. Results and Discussion

With an emphasis on performance measures and model size, this section displays the findings of the machine learning models assessed for surface categorization in the context of tractor surface identification. We also discuss the ramifications of these findings and how they apply to the real-world use of these models in embedded systems.

3.1. Performance Evaluation

Accuracy, macro precision, macro recall, macro F1 score, weighted precision, weighted recall, and weighted F1 score were among the performance indicators used to assess the models. These metrics offer a thorough assessment of the model's accuracy in classifying surface types while taking dataset imbalances and classification errors into account.

Table 2.
Performance Evaluation Results.

Model	Accuracy	Macro Precision	Macro Recall	Macro F1	Weighted Precision	Weighted Recall	Weighted F1
Logistic Regression	0.51	0.42	0.44	0.42	0.49	0.51	0.50
K Neighbors	0.86	0.84	0.84	0.84	0.86	0.86	0.86
SVC	0.73	0.69	0.69	0.69	0.75	0.73	0.73
Decision Tree	0.86	0.85	0.85	0.85	0.86	0.86	0.86
Random Forest	0.90	0.90	0.89	0.89	0.91	0.90	0.90
Gradient Boosting	0.86	0.85	0.84	0.84	0.86	0.86	0.86
AdaBoost	0.53	0.45	0.46	0.44	0.53	0.53	0.52
XGB Classifier	0.91	0.90	0.90	0.90	0.92	0.91	0.91

As shown in Table 2, the percentage of accurate forecasts among all predictions is known as accuracy. The models with the highest accuracy in our investigation were Random Forest and XGBoost, with 90% and 91%, respectively. This suggests that both models performed exceptionally well at differentiating across various surface types. With accuracies of about 86%, K-Nearest Neighbors (KNN) and Gradient Boosting also shown strong performance. However, models such as AdaBoost and Logistic Regression demonstrated comparatively lower accuracy ratings of 51% and 53%, respectively, indicating that they were less able to capture the intricacies in the data.

Particularly with unbalanced datasets, precision, recall, and F1 score provide a more detailed picture of model performance. The models that performed the best in these metrics were Random Forest and XGBoost, which showed balanced precision and recall for all surface types. With a macro F1 score of 0.90, XGBoost demonstrated a healthy trade-off between recall and precision. Further demonstrating its incapacity to effectively differentiate between surface types, Logistic Regression and AdaBoost fared badly in these metrics, exhibiting reduced precision and recall. The detailed Confusion Matrix of each model can be seen in Figure 7.

Figure 7.
Confusion Matrix Comparison of Each Model.

3.2. Model Size and Pruning

The model size is a crucial factor to take into account while deploying embedded systems. Reducing the size of machine learning models without compromising performance is crucial because of the memory and processing power limitations on devices like microcontrollers and IoT platforms.

Table 3.
Random Forest and XGBoost Model Size before and after Pruning.

File Name	Size (KB)
Original Random Forest.h	43,835 KB
Original XGB Classifier.h	3,893 KB
Pruned Random Forest.h	19,603 KB
Pruned XGB Classifier.h	102 KB

As shown in Table 3, the first versions of the Random Forest and XGBoost models were 43,835 KB and 3,893 KB, respectively. For embedded systems with small memory capacities, such microcontrollers, which normally have memory capacities between 32KB and 256KB, these sizes are impractical. The model sizes were greatly decreased during trimming. The size of the trimmed Random Forest and XGBoost models dropped to 19,603 KB and 102 KB, respectively. In particular, the XGBoost model's significant size reduction allows it to be deployed on devices with limited memory while retaining a high classification accuracy. This reduction in size was accomplished by restricting the number of trees in both the Random Forest and XGBoost models, eliminating redundant nodes, and pruning less important trees in the ensemble (Random Forest). The model became more effective for real-time inference on embedded systems due to the pruning strategies, which removed overfitting by deleting less crucial model parameters.

3.3. Model Suitability for Embedded Systems

XGBoost was shown to be the best model for embedded system deployment when performance and model size were taken into account. The trimmed XGBoost model was perfect for TinyML applications with limited memory and processing power because it was much smaller (102 KB) and showed high classification accuracy (91%). Random Forest, on the other hand, has a very large trimmed model size (19,603 KB), which could be an issue for devices with extremely limited resources, even though it offers comparable speed.

Despite having a respectably high accuracy of 86%, the KNN model might not be the best choice for embedded systems because it lacks built-in model size reduction strategies. Compared to models like XGBoost or Random Forest, where only the learnt parameters are needed for inference, KNN is less effective for deployment in embedded devices because it depends on keeping complete training data for classification.

Feature significance analysis offers important information about which sensor values are most important for surface categorization, especially for models like Random Forest and XGBoost. Gyroscope data, such as gyro-Z, and accelerometer readings, like acc-Z and acc-Y, were the main factors influencing model choices. These characteristics had the greatest influence on classification performance, suggesting that rotational movement along particular axes and acceleration are important surface type differentiators.

In real-world applications, where comprehending the model's decision-making process is essential for troubleshooting or enhancing model performance, the interpretability of these models is especially significant. Both XGBoost and Random Forest offer feature relevance scores, which let users see which features influence the model's choices. This interpretability helps with model refinement, especially when some features may need to be improved or better preprocessed.

The study's findings have applications in the implementation of machine learning models in agricultural machinery, including self-driving tractors. For real-time surface recognition, XGBoost presents a viable option, especially in situations with limitations like embedded devices. One significant benefit in the context of driverless cars or Internet of Things devices is the capacity to trim and shrink models without compromising accuracy.

With an emphasis on performance and model size for embedded deployment, this study assessed a number of machine learning models for surface classification in tractors. The most successful model was XGBoost, which achieved high classification accuracy and a small model size, making it appropriate for use in IoT devices with limited resources. Models were successfully pruned to minimize their size, especially for XGBoost, which made them suitable for TinyML applications. For effective implementation in operational contexts, future work can concentrate on further improving these models to improve their real-time performance and further minimize their memory footprint.

```
Sample 1903 - Predicted: 6, Actual: 6
Sample 1904 - Predicted: 6, Actual: 6
Sample 1905 - Predicted: 2, Actual: 0
Sample 1906 - Predicted: 0, Actual: 0
Sample 1907 - Predicted: 3, Actual: 3
Sample 1908 - Predicted: 3, Actual: 5
Sample 1909 - Predicted: 6, Actual: 6
Sample 1910 - Predicted: 6, Actual: 6
Sample 1911 - Predicted: 1, Actual: 1
Sample 1912 - Predicted: 5, Actual: 5
Sample 1913 - Predicted: 6, Actual: 6
Sample 1914 - Predicted: 3, Actual: 3
Sample 1915 - Predicted: 5, Actual: 5
Sample 1916 - Predicted: 1, Actual: 6
Sample 1917 - Predicted: 6, Actual: 6
Sample 1918 - Predicted: 3, Actual: 3
Sample 1919 - Predicted: 1, Actual: 1
Sample 1920 - Predicted: 2, Actual: 2
Sample 1921 - Predicted: 2, Actual: 2
Sample 1922 - Predicted: 1, Actual: 1
Sample 1923 - Predicted: 6, Actual: 6
Sample 1924 - Predicted: 6, Actual: 6
Sample 1925 - Predicted: 2, Actual: 4
Sample 1926 - Predicted: 5, Actual: 5
Sample 1927 - Predicted: 1, Actual: 0
Sample 1928 - Predicted: 5, Actual: 3
Sample 1929 - Predicted: 6, Actual: 6
```

Figure 8.
XGBoost Model deployment results on ESP32.

According to the ESP32 surface categorization deployment data as Figure 8, the XGBoost model's overall accuracy is 82.43%. This shows that the model accurately identified the surface type out of all the predictions it generated. A range of projected and actual surface types for individual samples are included in the results, which demonstrate how well the deployed model performed in various test scenarios.

With several accurate forecasts (e.g., Samples 1903, 1909, 1913, 1919, 1923, 1929), the model seems to work well on some surface types, such as Class 6. Misclassifications are also noted, though, especially in the instance of Class 0, where Sample 1906 is mistakenly projected as 0 rather than the true Class 3. Misclassifications indicate places where the model may be having trouble differentiating between specific surface types, such as when it predicts Class 1 when the actual class is 6 (e.g., Sample 1916).

The model's overall accuracy of 82.43% indicates that it is generally reliable but still has potential for development, especially when it comes to differentiating across classes that may have similar characteristics. Particularly in embedded contexts like ESP32, where CPU resources are constrained, extra fine-tuning, feature selection or even investigating more sophisticated model optimization techniques could increase accuracy and lower misclassification rates.

4. Conclusion

This research applied surface recognition in autonomous tractors by using machine learning models. The results demonstrate percentage accuracy of surface recognition. The models with the highest accuracy in our investigation were Random Forest and XGBoost, with 90 percent and 91 percent, respectively. This suggests that both models performed exceptionally well at differentiating across various surface types. With accuracies of about 86 percent, K-Nearest Neighbors (KNN) and Gradient Boosting also have shown strong performance. However, models such as AdaBoost and Logistic Regression demonstrated comparatively lower accuracy ratings of 51 percent and 53 percent, respectively, indicating that they were less able to capture the intricacies in the data. Particularly with unbalanced datasets, precision, recall, and F1 score provide a more detailed picture of model performance. The models that performed the best in these metrics were Random Forest and XGBoost, which showed balanced precision and recall for all surface types. With a macro F1 score of 0.90, XGBoost demonstrated a healthy trade-off between recall and precision. Further demonstrating its incapacity to effectively differentiate between surface types, Logistic Regression and AdaBoost fared poorly in these metrics, exhibiting reduced precision and recall. In order to provide a thorough foundation for practical precision agricultural applications, ongoing research will concentrate on assessing the effects of pruning and quantization on inference speed and accuracy on real embedded hardware.

References

[1] P. V. Borges *et al.*, "A survey on terrain traversability analysis for autonomous ground vehicles: Methods, sensors, and challenges," *Field Robotics*, vol. 2, pp. 1567-1627, 2022. http://doi:10.55417/fr.2022049

[2] G. Aiello, P. Catania, M. Vallone, and M. Venticinque, "Worker safety in agriculture 4.0: A new approach for mapping operator's vibration risk through Machine Learning activity recognition," *Computers and Electronics in Agriculture*, vol. 193, p. 106637, 2022. http://doi:10.1016/j.compag.2021.106637

[3] F. K. Shaikh, S. Karim, S. Zeadally, and J. Nebhen, "Recent trends in internet-of-things-enabled sensor technologies for smart agriculture," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23583-23598, 2022.

[4] A. N. Wilson, A. Kumar, A. Jha, and L. R. Cenkeramaddi, *Embedded sensors, communication technologies, computing platforms and machine learning for UAVs: A review*. United States: Institute of Electrical and Electronics Engineers Inc, 2022.

[5] F. Palermo *et al.*, "Advancements in context recognition for rdge devices and smart eyewear: Sensors and applications," *IEEE Access*, pp. 1-39, 2025. http://doi:10.1109/ACCESS.2025.3555426

[6] X. Shan *et al.*, "Low-resource scenario classification through model pruning towards refined edge intelligence," *IEEE Internet of Things Journal*, vol. 11, no. 21, pp. 34398-34408, 2023. http://doi:10.1109/JIOT.2023.3347665

[7] P. Thavitchasri, D. Maneetham, and P. N. Crisnapati, "Intelligent surface recognition for autonomous tractors using ensemble learning with BNO055 IMU sensor data," *Agriculture*, vol. 14, no. 9, p. 1557, 2024. http://doi:10.3390/agriculture14091557

[8] M. M. Kabir, J. R. Jim, and Z. Istenes, "Terrain detection and segmentation for autonomous vehicle navigation: A state-of-the-art systematic review," *Information Fusion*, vol. 113, p. 102644, 2025. http://doi:10.1016/j.inffus.2024.102644

[9] E. Cumbajin *et al.*, "A systematic review on deep learning with CNNs applied to surface defect detection," *Journal of Imaging*, vol. 9, no. 10, p. 193, 2023. http://doi:10.3390/jimaging9100193

[10] K. Sharma and S. K. Shivandu, "Integrating artificial intelligence and internet of things (IoT) for enhanced crop monitoring and management in precision agriculture," *Sensors International*, p. 100292, 2024. http://doi:10.1016/j.sintl.2024.100292

[11] E. Njor, M. A. Hasanpour, J. Madsen, and X. Fafoutis, "A holistic review of the TinyML stack for predictive maintenance," *IEEE Access*, pp. 184861-184882, 2024. http://doi:10.1109/ACCESS.2024.3512860

[12] L. Dutta and S. Bharali, "Tinyml meets iot: A comprehensive survey," *Internet of Things*, vol. 16, p. 100461, 2021. http://doi:10.1016/j.iot.2021.100461

[13] S. Buya, P. Tongkumchum, and B. E. Owusu, "Modelling of land-use change in Thailand using binary logistic regression and multinomial logistic regression," *Arabian Journal of Geosciences*, vol. 13, pp. 1-12, 2020. http://doi:10.1007/s12517-020-05451-2/Published

[14] Y. Chen *et al.*, "Fast density peak clustering for large scale data based on kNN," *Knowledge-Based Systems*, vol. 187, p. 104824, 2020. http://doi:10.1016/j.knosys

[15] A. N. Beskopylny *et al.*, "Concrete strength prediction using machine learning methods CatBoost, k-nearest neighbors, support vector regression," *Applied Sciences*, vol. 12, no. 21, p. 10864, 2022. https://doi.org/10.3390/app122110864

[16] A. Hashemizadeh, A. Maaref, M. Shateri, A. Larestani, and A. Hemmati-Sarapardeh, "Experimental measurement and modeling of water-based drilling mud density using adaptive boosting decision tree, support vector machine, and K-nearest neighbors: A case study from the South Pars gas field," *Journal of Petroleum Science and Engineering*, vol. 207, p. 109132, 2021. http://doi:10.1016/j.petrol.2021.109132

[17] Y. Yaguchi and K. Tamagawa, "A waypoint navigation method with collision avoidance using an artificial potential method on random priority," *Artificial Life and Robotics*, vol. 25, no. 2, pp. 278-285, 2020. http://doi:10.1007/s10015-020-00583-w

[18] F. Huber, A. Yushchenko, B. Stratmann, and V. Steinhage, "Extreme gradient boosting for yield estimation compared with Deep learning approaches," *Computers and Electronics in Agriculture*, vol. 202, p. 107346, 2022. http://doi:10.1016/j.compag.2022.107346

[19] Y. Li, Y. Guo, L. Gong, and C. Liu, "Harvesting route detection and crop height estimation methods for lodged farmland based on AdaBoost," *Agriculture*, vol. 13, no. 9, p. 1700, 2023. http://doi:10.3390/agriculture13091700

[20] D. A. L. Mariadass, E. G. Moung, M. M. Sufian, and A. Farzamnia, "Extreme gradient boosting (XGBoost) regressor and shapley additive explanation for crop yield prediction in agriculture," presented at the 2022 12th International Conference on Computer and Knowledge Engineering, ICCKE 2022, Institute of Electrical and Electronics Engineers Inc, 2022.

[21] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. Van De Weijer, "Class-incremental learning: survey and performance evaluation on image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5513-5533, 2022. http://doi:10.13039/501100011033

[22] A. T. Nguyen, Y. Ahn, S. Park, S. Park, and D. H. Pham, "Meta learning regression framework for energy consumption prediction in retrofitted buildings: A case study of South Korea," *Journal of Building Engineering*, vol. 96, p. 110403, 2024. http://doi:10.1016/j.jobe.2024.110403

[23] K. I. Sundus, B. H. Hammo, M. B. Al-Zoubi, and A. Al-Omari, "Solving the multicollinearity problem to improve the stability of machine learning algorithms applied to a fully annotated breast cancer dataset," *Informatics in Medicine Unlocked*, vol. 33, p. 101088, 2022. http://doi:10.1016/j.imu.2022.101088

[24] Y. Wang and Y. Li, "Mapping the ratoon rice suitability region in China using random forest and recursive feature elimination modeling," *Field Crops Research*, vol. 301, p. 109016, 2023. http://doi:10.1016/j.fcr.2023.109016

[25] A. Aloisio, M. M. Rosso, A. M. De Leo, M. Fragiacomo, and M. Basi, "Damage classification after the 2009 L'Aquila earthquake using multinomial logistic regression and neural networks," *International Journal of Disaster Risk Reduction*, vol. 96, p. 103959, 2023. http://doi:10.1016/j.ijdrr.2023.103959

[26] S. Memiş, S. Enginoğlu, and U. Erkan, "Fuzzy parameterized fuzzy soft k-nearest neighbor classifier," *Neurocomputing*, vol. 500, pp. 351-378, 2022. http://doi:10.1016/j.neucom.2022.05.041

[27] W. Tang, "Application of support vector machine system introducing multiple submodels in data mining," *Systems and Soft Computing*, vol. 6, p. 200096, 2024. http://doi:10.1016/j.sasc.2024.200096

[28] I. D. Mienye and N. Jere, "A survey of decision trees: Concepts, algorithms, and applications," *IEEE Access*, vol. 12, pp. 86716–86727, 2024. http://doi:10.1109/ACCESS.2024.3416838

[29] M. M. Ghiasi and S. Zendehboudi, "Application of decision tree-based ensemble learning in the classification of breast cancer," *Computers in Biology and Medicine*, vol. 128, p. 104089, 2021. http://doi:10.1016/j.compbiomed.2020.104089

[30] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *Institute of Electrical and Electronics Engineers Inc*, vol. 10, pp. 99129-99149, 2022. http://doi:10.1109/ACCESS.2022.3207287

[31] N. Javaid, M. Akbar, A. Aldegheishem, N. Alrajeh, and E. A. Mohammed, "Employing a machine learning boosting classifiers based stacking ensemble model for detecting non technical losses in smart grids," *IEEE Access*, vol. 10, pp. 121886-121899, 2022. http://doi:10.1109/ACCESS.2022.3222883

[32] A. I. A. Osman, A. N. Ahmed, M. F. Chow, Y. F. Huang, and A. El-Shafie, "Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 1545-1556, 2021. http://doi:10.1016/j.asej.2020.11.011

[33] H. Wang *et al.*, "A comprehensive survey on training acceleration for large machine learning models in IoT," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 939-963, 2021. http://doi:10.1109/JIOT.2021.3111624

[34] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-20, 2024. http://doi:10.1109/TPAMI.2024.3447085

[35] T. Suwannaphong, F. Jovan, I. Craddock, and R. McConville, "Optimising tinyML with quantization and distillation of transformer and mamba models for indoor localisation on edge devices," Retrieved: https://arxiv.org/abs/2412.09289, 2024.

[36] A. Nanda, "Quantizing the weights of AI models. Towards Data Science," Retrieved: https://medium.com/towards-data-science/quantizing-the-weights-of-ai-models-39f489455194, 2023.

[37] J. Lee, H. Kim, and S. Park, "Efficient machine learning for embedded systems: Optimizing models for real-time applications in autonomous vehicles," *Journal of Embedded Systems*, vol. 17, no. 2, pp. 104-116, 2023.