World Journal of Engineering Research and Technology

www.wjert.org

Impact Factor: 7.029 Coden USA: WJERA4

APPLICATION OF MACHINE LEARNING ALGORITHMS IN LIDAR POINT CLOUD CLASSIFICATION

Trung Nguyen Tu*[1], Tran Thi Thu Ngan[2]

[1]Faculty of Computer Science and Engineering, Thuyloi University, 175 Tay Son, Dong Da, Hanoi, 010000, Vietnam.

[2]Faculty of Technology and Data Science, Foreign Trade University, 91 Chua Lang Street, Lang Thuong Ward, Dong Da District, Hanoi, Vietnam.

Article Received on 18/03/2025 **Article Revised on 07/04/2026** **Article Published on 01/05/2026**

***Corresponding Author**

Trung Nguyen Tu

Faculty of Computer Science and Engineering, Thuyloi University, 175 Tay Son, Dong Da, Hanoi, 010000, Vietnam.

https://doi.org/10.5281/zenodo.19886044

How to cite this Article: Trung Nguyen Tu*[1], Tran Thi Thu Ngan[2]. (2026). Application Of Machine Learning Algorithms In Lidar Point Cloud Classification. World Journal of Engineering Research and Technology, 12(4), 57–68.

ABSTRACT

The extraction and classification of features from three-dimensional (3D) LiDAR point clouds have become essential in photogrammetry, remote sensing, and smart city modeling. This study investigates the application of a number of supervised machine learning methods for classifying LiDAR point clouds, including Random Forest, Decision Tree, Naïve Bayes, Neural Networks, AdaBoost, Support Vector Machines (SVM), and K-Nearest Neighbours (K-NN). Experiments were conducted on two datasets with various geometric and radiometric features. Evaluation measures such F1-Score, Precision, and Recall, and computational time were analyzed. The results demonstrate the effectiveness of integrating geometric features and ensemble learning strategies in improving classification performance.

KEYWORDS: LiDAR; point cloud; machine learning; features, classification.

1. INTRODUCTION

In recent years, the construction of 3D city models has become a critical research area that supports a wide range of applications, including urban surface analysis, infrastructure planning, environmental monitoring, and smart city initiatives. These models are often created by

integrating multiple data sources such as digital elevation models (DEM), satellite and aerial imagery, topographic maps, photogrammetric data, and particularly LiDAR point clouds (Bui et al., 2020; Dương et al., 2022; Lê, 2019, 2023). LiDAR has gained prominence because it provides high-density three-dimensional representations of terrain and built environments, offering a rich basis for precise 3D reconstruction.

LiDAR point clouds contain millions of spatially distributed points that not only represent XYZ coordinates but also often include intensity and color information. Processing these large, unstructured datasets to extract meaningful objects—such as buildings, vegetation, and ground surfaces—through classification remains challenging due to irregular point distributions, varying densities, and noise.

Traditional approaches often rely on heuristic or rule-based methods, such as morphological filters or manually crafted geometric criteria. However, these techniques struggle to adapt to diverse and complex urban scenes. Machine learning has therefore emerged as a robust alternative, learning classification rules directly from labeled data and reducing the dependence on manually tuned parameters (Weinmann et al., 2015; Guo et al., 2019).

Conventional machine learning techniques including Naïve Bayes, Random Forests, Decision Trees, Support Vector Machines, and K-Nearest Neighbors—have been widely adopted to classify LiDAR point clouds using geometric features computed from local neighborhoods. Ensemble methods like Random Forest and boosting approaches have shown particular promise, effectively combining multiple weak learners to improve generalization and mitigate overfitting (Alexandre, 2020; Cabo et al., 2019; Atik et al., 2021).

Beyond traditional approaches, the rise of deep learning has revolutionized point cloud processing. Models such as PointNet and PointNet++ can directly learn from raw point clouds, extracting hierarchical features while maintaining spatial relationships (Qi et al., 2017). Graph-based methods like DGCNN further capture local geometric structures by dynamically updating neighborhood graphs (Wang et al., 2019).

More recently, research in 2024 and 2025 has focused on incorporating transformer architectures and multi-modal fusion to improve classification performance on LiDAR data. Studies such as those on BiasFormer and 3DLST have introduced attention mechanisms tailored to handle uneven point distributions and learn robust features across different scales.

The Camera-LiDAR Fusion Transformer (CLFT) developed in 2025 highlights the trend of combining image and LiDAR data to better recognize complex urban scenes. Meanwhile, the USGS has deployed automated transformer-based pipelines to classify large-scale 3DEP LiDAR datasets, supporting national DEM generation with improved accuracy.

Despite these advancements, challenges remain in classifying LiDAR data from areas with intricate topographies or heterogeneous land covers. This study therefore systematically evaluates a suite of machine learning strategies—spanning from classical algorithms to the latest ensemble and transformer-based models—and integrates geometric and radiometric features to enhance classification accuracy. Through this work, we aim to advance methodologies that support robust and scalable 3D city modeling and terrain analysis for future smart urban development.

2. METHODS

2.1. Overview of LiDAR

One active remote sensing method is LiDAR (Light Detection and Ranging) that determines distances by sending laser pulses toward targets and measuring the time taken for the reflections to return. This process results in the creation of dense three-dimensional (3D) point clouds, capturing precise spatial structures of terrain, vegetation, and built environments (Wehr & Lohr, 1999; Shan & Toth, 2018).

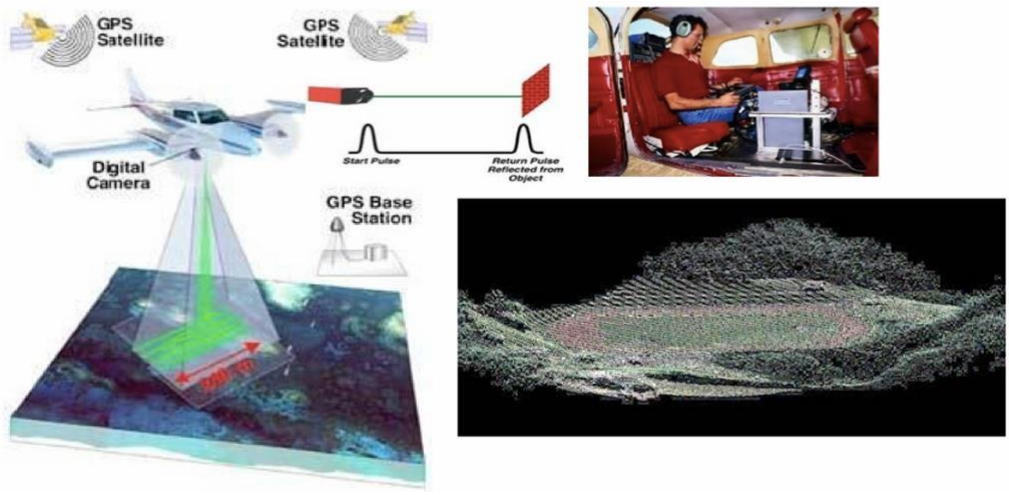

Figure 1: Overview of LiDAR system.

Compared to traditional photogrammetric approaches, LiDAR directly provides elevation data independent of lighting conditions, and can penetrate sparse vegetation canopies, making it invaluable for applications such as digital elevation modeling, flood risk analysis, forestry

inventory, and urban infrastructure mapping (Mallet & Bretar, 2009; Vosselman & Maas, 2010).

Airborne LiDAR systems are widely used to capture large-area topography, while terrestrial and mobile LiDAR platforms enable high-resolution scans of streetscapes and buildings, essential for engineering, cultural heritage documentation, and autonomous navigation (Sithole & Vosselman, 2004).

Nevertheless, raw LiDAR point clouds are inherently unstructured, often containing noise and multiple returns from different object layers. This underscores the need for advanced processing techniques, especially classification methods, to effectively separate ground surfaces from vegetation and man-made structures (Weinmann et al., 2015). Recently, machine learning approaches have demonstrated remarkable success in automating LiDAR data interpretation, driving more accurate and efficient 3D modeling for smart city applications (Duran et al., 2021; Ciou et al., 2024).

2.2. LiDAR Point Cloud Classification Problem

LiDAR point cloud classification is the process of assigning each point in a 3D dataset acquired by LiDAR scanning (Light Detection and Ranging) to categories such as ground, vegetation, buildings, or other structures Shan & Toth (2018). The main goal is to automatically separate and identify surface types based on geometric, optical, and spatial characteristics, serving applications such as:

- Digital terrain modeling (DTM, DEM)
- Urban monitoring (smart city)
- Infrastructure planning
- Forest inventory or natural disaster risk analysis

2.2. Structure of LAS File

The American Society for Remote Sensing and Photogrammetry(ASPRS) created the LAS file format, a public binary standard, which is frequently used to store LiDAR point cloud data. The LAS format efficiently organizes 3D point data along with associated attributes, enabling interoperability across various LiDAR processing software (ASPRS, 2013).

A typical LAS file is structured into several key sections:

- File Header: Contains global information about the dataset such as the point count, spatial

reference system, bounding box coordinates (min/max X, Y, Z), and versioning.

- Variable Length Records (VLRs): Store additional metadata, such as projection details (WKT strings or EPSG codes), waveform data descriptors, or custom application-specific tags.
- Point Data Records: The core section holding individual LiDAR points. Each record includes:

- 3D coordinates (X, Y, Z) stored as scaled integers.
- Intensity value (reflectance of the laser return).
- Return number and total returns (helping identify multiple returns from the same pulse).
- Classification flags (e.g., ground, vegetation, building).
- Scan angle, point source ID, and optionally GPS time.
- Extended versions (1.3 onward) can include RGB color, near-infrared (NIR) values, and waveform packets.

- Extended Variable Length Records (EVLRs): Added in LAS 1.4, allowing larger metadata payloads beyond the original VLR limits.

Understanding the LAS structure is fundamental for pre-processing, filtering, and feeding meaningful attributes into machine learning pipelines for tasks such as classification or segmentation.

2.3. Machine Learning Overview

This section provides a detailed overview of key machine learning algorithms used in LiDAR point cloud classification, including their mathematical principles, strengths, and typical use cases (Shinohara et al., 2022; Weinmann, 2020).

- Naïve Bayes (NB): Based on Bayes' theorem (Bishop, 2006):

$$P(Cl_k|x) = \frac{P(x|Cl_k)P(Cl_k)}{P(x)} \quad (1)$$

Assumes independence among features. Often uses Gaussian distributions for continuous data (Bishop, 2006):

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}} \quad (2)$$

Simple and computationally efficient, suitable for large datasets but can be limited by its

independence assumption (Zhang et al., 2014).

- Decision Tree Classifier (DTC): Builds a tree by selecting splits that maximize information gain or Gini impurity (Quinlan, 1986):

$$IG(D,A) = H(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} H(D_v) \quad (3)$$

Easy to interpret and visualize, though prone to overfitting on noisy data (Breiman et al., 1984).

- Random Forest (RF): An ensemble of decision trees using bagging and feature randomness. Final prediction by majority voting (Breiman, 2001):

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_m(x)\} \quad (4)$$

Highly robust, reduces overfitting, and handles large feature spaces well (Breiman, 2001; Zhang et al., 2018).

- AdaBoost (ADB): Sequentially trains weak learners, updating weights (Freund & Schapire, 1997):

$$w_i^{(t+1)} = w_i^{(t)} e^{-\alpha_t y_i h_t(x_i)}, \quad \alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \quad (5)$$

Focuses learning on difficult samples, improves weak classifiers into a strong ensemble (Freund & Schapire, 1997).

- K-Nearest Neighbors (K-NN): classifies the K closest samples according to the majority class, using distance (often Euclidean) (Cover & Hart, 1967):

$$d(p, p_i) = \sqrt{\sum_{j=1}^{n} (p_j - p_{ij})^2} \quad (6)$$

Non-parametric, simple to implement but computationally intensive on large datasets (Gualtieri et al., 2016).

- Support Vector Machines (SVM): Finds hyperplane maximizing the margin between classes (Vapnik, 1995):

$$\max_{w,b} \frac{2}{||w||}, \quad \text{s.t. } y_i(w^T x_i + b) \geq 1 \quad (7)$$

Kernel functions (e.g., RBF, polynomial) extend SVMs to non-linear boundaries (Schölkopf et al., 1998), making them powerful for complex datasets.

- Neural Networks (MLP): Consist of multiple layers computing:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}, \quad a^{(l)} = \sigma(z^{(l)}) \qquad (8)$$

Trained via backpropagation (Rumelhart et al., 1986), they capture intricate non-linear relationships but require careful tuning and large datasets (Goodfellow et al., 2016).

3. Input Features

The local geometry of LiDAR points is commonly described using geometric features. The geometric link between a point and the points in its immediate vicinity is established by these characteristics. These geometric traits can be identified based on a given scale. Three categories are used to derive the geometric attributes of any point p in the point cloud:

- Z-Z_{min} and Intensity
- (XYZ)-$(XYZ)_{min}$ and Intensity
- (XYZ)-$(XYZ)_{min}$, Intensity and RGB

4. Experiments

4.1. Settings and Data

The machine learning algorithms used in the experiments include: Naïve Bayes, Decision Tree, Random-Forest, Neural-Net, AdaBoost, Nearest-Neighbors, SVM. Datasets and Classes are showed in table 1 and 2.

Table 1: The datasets used are described in the following table.

STT	Desciption	Train	Test
1	Data 1	2,789,468 points	697,367 points
2	Data 2	7,100,628 points	1,775,156 points

Table 2: List of classes.

STT	Name En	Name Vn
1	Ground	Mặt đất
2	Vegetation	Thực vật
3	Bulding	Tòa nhà
4	Road	Đường bộ
5	Conductor	Dây dẫn điện
6	Shield Wire	Dây chống sét
7	Structure	Cột điện
8	Insulator	Thiết bị cách điện
9	Drainage Thread	Dây thoát nước

4.2. Evaluation indicators

The evaluation measures, which are computed as follows, comprise F1-score, Precision, and

Recall:

4.3. Experimental Results

4.3.1. Experiment 1

$$Precision = \frac{TP}{FP+TP} \quad (8)$$

$$Recall = \frac{TP}{FN+TP} \quad (9)$$

$$F1 = \frac{Recall*Precision*2}{Recall+Precision} \quad (10)$$

Table 3: Results with Features: Z-Z_{min} and Intensity.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.7166	0.6783	0.6728
Decision tree	0.7635	0.8097	0.7386
Random forest	0.7727	0.7237	0.7411
Nearest-neighbors	0.7743	0.7875	0.7795
Neural-net	0.7384	0.8075	0.7483
Adaboost	0.6473	0.7991	0.7151
SVM	0.5970	0.0681	0.0431

Table 4: Results with Features: (XYZ)-$(XYZ)_{min}$ and Intensity.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.7834	0.7138	0.7249
Decision tree	0.8477	0.8659	0.8410
Random forest	0.9438	0.9173	0.9248
Nearest-neighbors	0.9285	0.9274	0.9279
Neural-net	0.6804	0.8064	0.7330
Adaboost	0.7231	0.5651	0.5969
SVM	0.6257	0.3744	0.4486

Table 5: Results with Features: (XYZ)-$(XYZ)_{min}$, Intensity and RGB.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.7935	0.7546	0.7665
Decision tree	0.8566	0.8744	0.8493
Random forest	0.937	0.9201	0.9254
Nearest-neighbors	0.9147	0.916	0.9153
Neural-net	0.6909	0.8163	0.7458
Adaboost	0.7231	0.5651	0.5969
SVM	0.7161	0.6109	0.6421

4.3.2. Experiment 2

Table 6: Results with Features: Z-Z_{min} and Intensity.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.6896	0.7528	0.7145

Decision tree	0.7703	0.8363	0.7981
Random forest	0.8091	0.7706	0.7865
Nearest-neighbors	0.8135	0.8233	0.8179
Neural-net	0.7432	0.8279	0.7819
Adaboost	0.6768	0.7605	0.7027
SVM	0.5738	0.0462	0.0529

Table 7: Results with Features: (XYZ)-$(XYZ)_{min}$ and Intensity.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.7087	0.7631	0.7281
Decision tree	0.8965	0.9058	0.8795
Random forest	0.975	0.9695	0.9711
Nearest-neighbors	0.9726	0.9709	0.9716
Neural-net	0.7611	0.8569	0.8058
Adaboost	0.4828	0.6412	0.5478
SVM	0.6969	0.4155	0.4698

Table 8: Results with Features: (XYZ)-$(XYZ)_{min}$, Intensity and RGB.

Algorithm	Precision	Recall	F1-Score
Naive bayes	0.8012	0.8196	0.8024
Decision tree	0.8962	0.9128	0.8941
Random forest	0.9744	0.9721	0.9729
Nearest-neighbors	0.9463	0.9484	0.9471
Neural-net	0.9077	0.9209	0.9070
Adaboost	0.8127	0.3931	0.5216
SVM	0.6787	0.4006	0.4662

4.4. Comments of Experimental Results

4.4.1. About Features

- When using only Z-Zmin and Intensity, the overall Precision, F1-Score and Recall of the models are quite low, ranging from 0.7 to 0.8 for good models, except for SVM (usually very low, almost inactive).
- When adding X, Y (ie spatial location), all indexes improve significantly. In particular, Random Forest, Nearest-Neighbors achieve F1-Score above 0.92 or even up to 0.97, showing that spatial location is a very important feature.
- When adding RGB, with both datasets, the algorithms still maintain or slightly improve the results. For example: Neural-net on data 2 achieves F1-Score ~ 0.907, Random Forest ~ 0.9729, proving that combining color information helps the model discriminate better.

4.4.2. About Algorithms

- Random Forest and Nearest-Neighbors consistently outperform other models, especially

when full features are available. They achieve F1-Score above 0.97 when using (X-Xmin, Y-Ymin, Z-Zmin, Intensity, RGB), showing that they are very suitable for LiDAR classification.

- Decision Tree also performs quite well, often achieving F1-Score > 0.85 with full features.
- Neural Network gives average results, usually lower than Random Forest by ~0.1, but still good and stable, especially with RGB information.
- Naïve Bayes, Adaboost, Quadratic Discriminant maintain average results, usually F1-Score ~0.7-0.8, not too outstanding.
- SVM and its variants (Linear, Gamma) generally perform poorly, with many cases of F1-Score < 0.05 or only slightly better than Gamma when there are many features, still < 0.7.

5. CONCLUSIONS

This study systematically evaluated the application of supervised machine learning algorithms—ranging from Naïve Bayes, Decision Tree, Random Forest, and K-NN to Neural Networks—for LiDAR point cloud classification. Experiments on two large datasets showed that adding spatial features ($X\text{-}X_{min}$,$Y\text{-}Y_{min}$), height (Z-Zmin), intensity, and RGB significantly boosted accuracy. Random Forest and K-NN achieved F1-Scores exceeding 0.97, while SVM performed poorly. This underscores the importance of feature selection and ensemble methods in LiDAR classification. Future work suggests integrating transformers and multi-modal learning (combining LiDAR and imagery) to further improve accuracy and scalability.

REFERENCES

1. Alexandre, B. (2020). ConvPoint: Continuous Convolutions for Point Cloud Processing. Computers & Graphics.
2. Atik, M. E., et al. (2021). Performance evaluation of machine learning methods for LiDAR point cloud classification. Drones.
3. Bui, N. Q., et al. (2020). Method of defining parameters for UAV point cloud classification algorithm.
4. Cabo, C., et al. (2019). Multiscale supervised classification of point clouds with urban and forest applications. Sensors.
5. Dương, A. Q., et al. (2022). Integrated approaches in 3D city modeling from LiDAR data.
6. Guo, Y., et al. (2019). Deep learning for 3D point clouds: A survey. IEEE TPAMI.

7. Lê, D. H. (2019, 2023). Studies on smart city 3D reconstruction.
8. Qi, C. R., et al. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. CVPR.
9. Wang, Y., et al. (2019). Dynamic Graph CNN for Learning on Point Clouds. ACM TOG.
10. Weinmann, M., et al. (2015). Semantic point cloud interpretation based on optimal neighborhoods. ISPRS Journal.
11. BiasFormer & 3DLST Papers (2024). Advanced transformer architectures for LiDAR point cloud classification.
12. USGS (2025). Automated transformer-based point cloud classification for national DEM.
13. CLFT (2025). Camera-LiDAR Fusion Transformer for multi-modal urban scene understanding.
14. Mitchell, T.M. (1997). Machine Learning. McGraw Hill.
15. Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1(1): 81-106.
16. Breiman, L. (2001). Random Forests. Machine Learning, 45(1): 5-32.
17. Freund, Y., & Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1): 119-139.
18. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1): 21-27.
19. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3): 273-297.
20. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning representations by back
21. Wehr & Lohr (1999) A. Wehr and U. Lohr, "Airborne laser scanning—an introduction and overview," ISPRS Journal of Photogrammetry and Remote Sensing, 1999; 54(2–3): 68–82.
22. Shan & Toth (2018) J. Shan and C. K. Toth (Eds.), Topographic Laser Ranging and Scanning: Principles and Processing, 2nd ed. Boca Raton, FL, USA: CRC Press, 2018.
23. Breiman, L. (2001). Random forests. Machine Learning, 45(1): 5–32.
24. Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees. Wadsworth International Group.
25. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
26. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1): 21–27.
27. Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning

and an application to boosting. Journal of Computer and System Sciences, 55(1): 119–139.

28. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
29. Gualtieri, J. A., et al. (2016). Performance of KNN on remote sensing data. Proceedings IGARSS.
30. Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1): 81–106.
31. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323: 533–536.
32. Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10(5): 1299–1319.
33. Shinohara, T., et al. (2022). A survey of machine learning for LiDAR data processing in autonomous driving. IEEE Transactions on Intelligent Transportation Systems, 23(2): 490–510.
34. Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer.
35. Weinmann, A. (2020). Machine learning on LiDAR point clouds and applications. Remote Sensing, 12(21): 3555.
36. Zhang, H., et al. (2018). Comparison of random forest and SVM classifiers for LiDAR point cloud classification. Remote Sensing, 10(2): 146.
37. Zhang, S., et al. (2014). Naïve Bayes classification of LiDAR features for vegetation. ISPRS Journal of Photogrammetry and Remote Sensing, 94: 87–98.