International Journal of
Environmental Sciences & Natural Resources
ISSN: 2572-1119

Juniper PUBLISHERS key to the Researchers

Review Article
Volume 32 Issue 3 - July 2023
DOI: 10.19080/IJESNR.2023.32.556340

Int J Environ Sci Nat Res

Classification of Forest LiDAR Data Using Deep Learning Pipeline Algorithm and Geometric Feature Analysis

Wijdan Amakhchan[1], Fayez Tarsha Kurdi[2]*, Zahra Gharineiat[2], Hakim Boulaassal[1] and Omar El Kharki[1]

[1]*GéoTéCa, FSTT, Abdelmalek Essaadi University, Morocco*

[2]*Department of Health, Engineering and Sciences, University of Southern Queensland, Australia*

Submission: June 16, 2023; **Published:** July 03, 2023

***Corresponding author:** Fayez Tarsha Kurdi School of Surveying and Built Environment, Faculty of Health, Engineering and Sciences, University of Southern Queensland, Springfield campus, QLD 4300, Australia

Abstract

This paper adapts the deep learning pipeline algorithm based on the Multi-Layer Perceptron (MLP) Neural Network to automatically classify the forest Light Detection And Ranging (LiDAR) point cloud. To achieve this, the Machine Learning (ML) algorithm parameters such as input layer elements, number of hidden layers, activation functions, and alpha value are optimized to achieve the best possible performance. Regarding the important role of the geometric features in the input layer, most of the suggested features in the literature are analyzed to employ the more effective ones in the algorithm input layer. As a result, seven geometric features, in addition to the 3D coordinates of the point cloud, are chosen to represent the first algorithm layer. The proposed algorithm classifies the forest LiDAR point cloud into two classes: vegetation and terrain. The proposed approach was tested using two points of clouds, one of a flat area and the other of a mountain area. The results of using the suggested approach provide an accuracy score greater than 98%. The obtained result confirms the high efficiency of the proposed classification algorithm regarding the envisaged approaches in the literature. Finally, the next step is to generalize this approach to classify more complicated scenes as urban areas.

Keywords: LiDAR; Classification; Vegetation; Forest areas; Machine learning

Introduction

Due to the data acquisition speed as well as the high data accuracy, Light Detection And Ranging (LiDAR) is becoming increasingly one of the major data sources of 3D spatial data [1]. In forest areas, according to Shan & Toth [2], the penetration of laser pulses through the canopy and measuring points under trees give additional importance to LiDAR data, especially for forest resource management, benefiting from advances in deep learning algorithms. In the last few years, the use of LiDAR technology in forest areas has grown, and it attracts researchers and companies to invest in it, especially after the climate changes that affect our environments. In which, forests play a critical role in its continuity and stability. LiDAR has been used for various applications in the forest, such as vegetation extraction, classification, and modeling.

In this context, Yang et al. [3] used LiDAR data for modeling the forest AboveGround Biomass (AGB). Their suggested approach is based on allometric relationships and the power-law form to integrate structural and spectral information by combining LiDAR canopy height attributes and optical spectral indexes. Loh et al. [4] investigated how multi-temporal airborne LiDAR data can be used to estimate the AGB changes in the tropical montane forest of Borneo. Jiang et al. [5] utilized spaceborne LiDAR to propose an optimized Extreme Learning Machine (ELM) method for estimating the AGB of natural forests on the eastern Qinghai-Xizang Plateau in China. In addition, Chen et al. [6] compared different stratification approaches, modeling algorithms, and categorical boosting to determine the best stratification approach and modeling algorithm for estimating forest Above Ground Carbon density (AGC) using airborne LiDAR data. While Fekry et al. [7] developed a general framework for integrating ground-based and drone mounted LiDAR data to improve tree parameter estimation using Quantitative Structure Modeling (QSM). Xiong et al. [8] measure the long-term structural changes in South Florida mangrove forests caused by Hurricane Irma in September 2017 using airborne LiDAR data.

Though ground and off the terrain is well-established research with documented success and a long list of developed algorithms and pipelines (see Section 2), both rule-based [9-13] and ML classification algorithms [14] still have limitations and need more improvement. That is why a long list of papers was published about the same topic during the last three years [15]. Indeed, the question of classification parameter selection manifests in rule-based approaches [16], whereas the classification accuracy in the ML classification algorithm still needs improvement (see Table 4 in Section 7). These motivations were behind deciding to achieve this research.

This paper proposes a new automatic classification approach for LiDAR point clouds in forest areas. The proposed algorithm aims to classify a forest LiDAR point cloud into two mains classes: vegetation and terrain. The suggested approach belongs to the ML approach family because it uses a pipeline of MultiLayers Perceptron (MLP) classifier directly on LiDAR data in addition to the point features, maximizing the model's prediction performance. At this stage, it is important to highlight the novelties as well as the importance of the suggested approach as follows:

a) Novel Deep Learning (DL) approach achieving the LiDAR point cloud classification with high accuracy.

b) Adaptation of given ML algorithm to minimize the classification errors.

c) Analysis of the point feature histograms to select the effective features that will be used with the point cloud as the input layer.

d) The high obtained classification accuracy is confirmed through the application of the suggested approach on different quality and topography datasets.

Before exposing the suggested approach and the achieved experiments, it is important to summarize the related works in the literature in the next section.

Related Works

In forest areas, LiDAR data may be classified into two main classes: vegetation and non-vegetation. The non-vegetation class represents several classes such as terrain, buildings, and powerlines. Regarding the minimal percentage of the non-terrain points among the non-vegetation class, it is acceptable to neglect them and focus only on the terrain and vegetation classes [3]. Nevertheless, there are two main classification approach families in literature: rule-based and ML approaches [16].

In the rule-based approach family, several approaches have been developed and improved to classify the trees, and their features. Blomley et al. [17] examined existing methods in terms of the geometrical scale of feature extraction (whole tree, within crown partitions, or within laser footprint). They deduced that features were obtained separately from the various scales. They aimed to use the within-tree-crown distribution of within-footprint signal characteristics as extra features to classify tree species using LiDAR attributes. Further, Budei et al. [18] compared the accuracy of MultiSpectral LiDAR (MSL) for identifying single tree species to standard discrete single wavelength LiDAR. In addition, Yang et al. [19] extracted multiple features from an integrated system that can simultaneously acquire hyperspectral information and LiDAR data from two areas with different tree species configurations and growing environments. To detect the contributions of different features in the pixel-wise tree species classification, they devised various schemes that were based on various groups of features and classifiers. Also, Yu et al. [20] examined the utility of multispectral Airborne Laser Scanning (ALS) data for detecting individual trees and classifying tree species using Multispectral Airborne Laser Scanning. To develop a single-sensor solution for forest mapping that can provide species-specific information needed for forest management and planning.

To detect individual trees, Ning et al. [21] proposed a coarse-to-fine individual tree detection method based on treetop points extraction and radius expansion from Mobile Laser Scanning (MLS) point cloud data. As well as, Luo et al. [22] used a novel top-down approach, for extracting individual trees from urban MLS point clouds, that combined embedded pointwise directions and detected tree centers. A Multi-level Self-Adaptive individual tree detection method (MSA) was developed by Hui et al. [23] for the coniferous forests to improve detection accuracy using airborne LiDAR, by combining the advantages of raster and point-based methods. Zhou et al. [24] compared a new trunk-based tree detection and localization approach to two approaches: DBSCAN-based (density-based spatial clustering of applications with noise) and height density-based approaches These approaches were evaluated using leaf-off LiDAR data from two Unmanned Aerial Vehicles (UAVs) and a Geiger mode system with varying point densities, geometric accuracies, and environmental complexities.

Recent literature tends to use ML and, more specifically, deep learning algorithms to improve the existing methods and to develop more accurate approaches. For the classification topic, Hell et al. [25] classified tree species and standing dead trees using two deep neural networks (DNNs): PointCNN (Point Convolutional Neural Networks) and 3DmFVNet. They investigated the two DNNs PointCNN and 3DmFV-Net for their use in tree species classification using airborne LiDAR data. Mizoguchi et al. [26] suggested a new method for tree species classification using depth images created from LiDAR point clouds of trunks using Convolutional Neural Networks (CNN). Marrs et al. [27] analyzed ML techniques for tree species identification, namely: neural networks, k-nearest neighbors, and Random Forest (RF). They used co-registered LiDAR and hyperspectral data to obtain high classification accuracies when differentiating between tree species. To classify tree species in 3D point clouds collected from complex forest scenes using LiDAR data, Zou et al. [28] presented a new voxel-based deep learning method to extract the individual tree based on the density of the point clouds, then represents low-level features using voxel-based rasterization. In addition, Nguyen et al. [29] used a Weighted Support Vector Machine (WSVM) classifier for tree species classification based on LiDAR data. They considered different weights for distinct classes of tree species to reduce the impact of the class imbalance distribution and different weights for different training samples based on their importance for the evaluated classification problem.

Regarding the methods used for tree detection, Tarsha Kurdi et al. [30] summarized the main approaches based on the ML Random Forest algorithm for tree detection. Schmohl et al. [31] used a 3D neural network for individual tree detection from 3D ALS point clouds. Windrim et al. [32] also used deep learning models to isolate individual trees, determine tree stem points, and build a segmented model of the main tree stem that includes tree height, diameter, taper, and sweep from airborne laser scanning (ALS) of forests. Corte et al. [33] utilized high-density GatorEye UAV-based LiDAR point clouds to test four different ML approaches, namely Support Vector Regression, Random Forest, Artificial Neural Networks, and Extreme Gradient Boosting,. They investigated these ML approaches for indirect estimation of individual tree dendrometry metrics such as diameter at breast height, total height, and timber volume. Chen et al. [34] evaluated a deep learning framework to segment individual tree crowns from UAV-based LiDAR data by processing directly forest point clouds from four forest types: nursery base, monastery garden, mixed forest, and defoliated forest. Luo et al. [35] proposed a deep learning network for detecting individual trees in complex forests using UAV-based laser scanning point clouds that consists of ground filtering and tree extraction and it's based on a multi-channel information interpretation.

The development of ML algorithms follows an expanding rate, and it improves frequently and rapidly. Recently, the use of automated ML (autoML) has been tackled and started to be applied to improve the accuracy of the suggested approaches in the literature. In this context, Olson & Moore [36] presented an open-source genetic programming-based AutoML system Tree-based Pipeline Optimization Tool (TPOT) v0.3, that reduces a series of feature preprocessors and ML models to maximize classification accuracy on a supervised classification task. Also, Zoller & Huber [37] implemented a mixture of a short survey on AutoML and an examination of frameworks for AutoML and hyperparameter optimization (HPO) on real data, they summarize and evaluate critical AutoML techniques and methods for each step in the creation of an ML pipeline. Feurer et al. [38] created a new AutoML system based on the scikit-learn that improves on existing AutoML methods by automatically considering the past performance on similar datasets. Olson et al. [39] demonstrated the efficiency of an open source TPOT in Python on a series of simulated and real-world benchmark data sets. They concluded that TPOT can create ML pipelines that outperform a basic ML analysis while requiring little to no user input or prior knowledge. Milutinovic et al. [40] provided a reference implementation as well as a universal framework for expressing end-to-end pipelines of differentiable, potentially probabilistic ML primitives. Nikitin et al. [41] suggested an approach that automates the creation of composite ML pipelines that were based on computational workflows composed of models and data operations. The method incorporated key concepts from both automated ML and workflow management systems. Furthermore, Xin et al. [42] examined the characteristics, components, and topologies of typical industrial-strength ML pipelines at various granularities. They introduced model graphlets, which are specialized data models for representing and reasoning about repeatedly run components in these ML pipelines. and recognize multiple rich opportunities for optimization based on traditional data management concepts.

Finally, it is important to highlight other applications of LiDAR data in forest areas. In this context, a novel proposed approach is presented by Tao et al. [43] for detecting fairy circles in intertidal salt marshes from UAV-based LiDAR data by considering the specular reflections and recovering the scanning geometry. LiDAR data was also used for fire risk evaluation by Gonzalez-Olabarria et al. [44] at the landscape level using spatially continuous information for forest management purposes in the region of Soria, they incorporated LiDAR derived information, a forest resources inventory, understory and canopy fuel modeling, and fire behavior simulation models.

Datasets

Regarding the application of all experiments and analyses on the selected datasets, it is crucial to introduce the employed datasets before entering the paper crux. four datasets are used in this paper. The first and the second datasets are taken from Elvis - Elevation, and Depth [45] (see Figure 1 & Table 1). In fact, the third point cloud is the extension of the first one. This data represents an airborne scan of Toowoomba-Lockyer Valley, Queensland, Australia. It was acquired in 2015 using a fixed-wing aircraft between the 18th of August and the 14th of December 2015. The Inertial Measurements Unit (IMU) and post-processed airborne Global Navigation Satellite System (GNSS) logs were used to generate the LiDAR point cloud from the waveform instrument data. The point density is equal to 2 point/m^2. The third point cloud is of the south mountain area in Queensland, Australia (see Figure 2). This point cloud was collected by a UAV platform in 2022. For this purpose, a DJI M300 RTK drone carrying the TrueView 515 LiDAR flew the payload. The mean flying height is 50m, the point density is 250 point/m^2 (Table 1). The last dataset is of Gisborne, New Zealand 2018-2020, it was captured for Gisborne District by aerial surveys and their subcontractors between 2018 and 2020. Data management and distribution is by Toitu Te Whenua Land Information New Zealand [46]. However, the four datasets represent forested areas. From the topographic viewpoint, the first two clouds are of a flat area, whereas the third one is of a mountain area. In fact, this selection has been adopted to check the efficiency of the suggested approach with several zones of different topography natures.

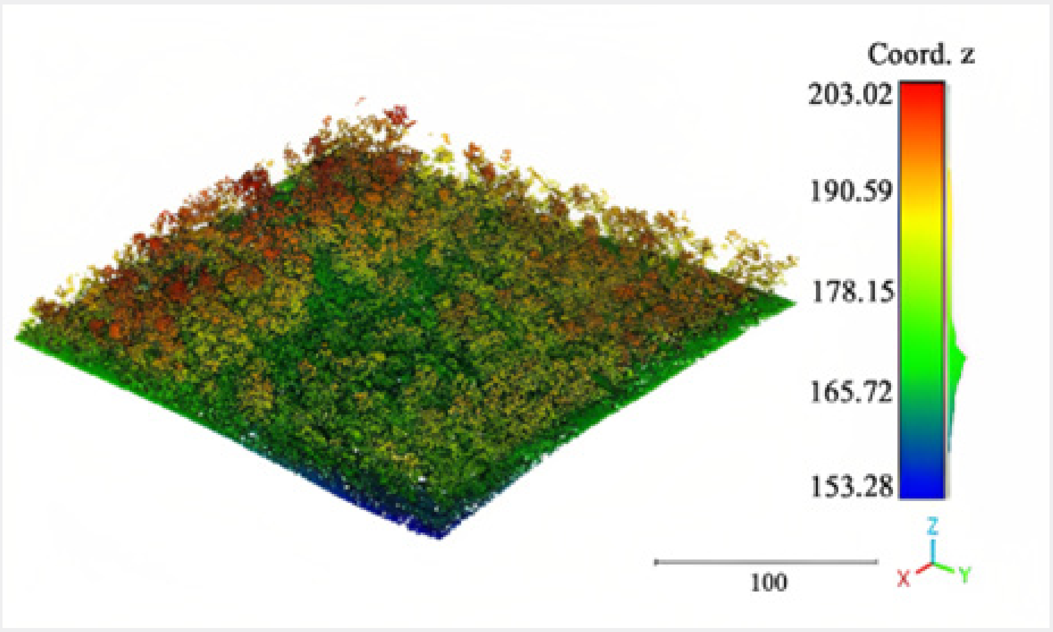

Figure 1: First dataset point cloud of the flat area (Table 1).

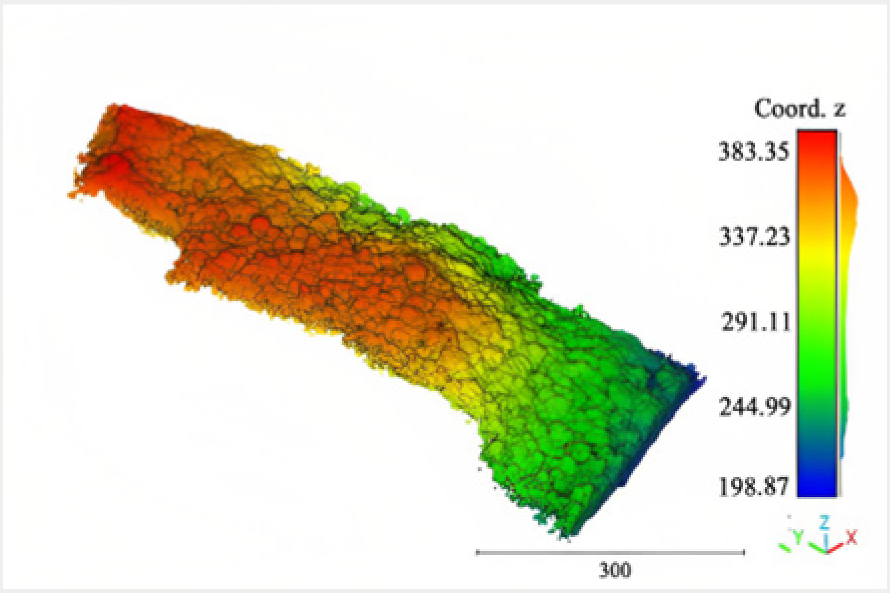

Figure 2: Third point cloud of the mountain area (Table 1).

Suggested Approach

This paper exposes a novel automatic classification approach for forest LiDAR datasets. The suggested approach classifies a forest LiDAR point cloud into two classes: vegetation and terrain. The input of the suggested approach is the LiDAR point cloud in addition to the calculated point features. According to west et al. [47], a list of geometric features can be calculated for each LiDAR point and its neighborhood. In this section, the components of the deep learning pipeline network will first be detailed. Thereafter, the geometric features of LIDAR points will be tested and analyzed to select the more effective feature that will be used with the LiDAR point cloud as the input layer in the proposed network.

Table 1: Characteristics of the input datasets.

	Area (ha)	Number of LAS Points	Flight Height (m)	Density (point/m^2)	Point Accuracy (cm)
Lockyer Valley North 2015 (Flat area)$^{-1}$	36	724 231	1600	2	30-80
Lockyer Valley North 2015-2	215	4343313	1600	2	30-80
Southeast of Queensland 2022 (Mountain area)	27	67 952 598	50	250	5-10
Gisborne, New Zealand 2018-2020	1498	237 622 088	--	10.07	---

Due to the large number of points in the mountain dataset, the point cloud was subsampled using the spatial method, where setting a minimum distance between two points is required, in this study 1 m was chosen. This means that the density of the new dataset is 1 point/m^2. The algorithm will be trained and tested with the subsampled dataset.

The next step is the labeling phase of the datasets, which involves selecting the items to be extracted. Since the purpose of this article is to extract vegetation, Cloudcompare software was used to visualize the points and manually separate the vegetation class from the other classes. When the separation process is complete, each class was assigned a number, “1” for the vegetation points and “0” for the others, which will be called in the rest of the paper terrain points. The labeled classes of each area were combined to obtain the final datasets that will train the algorithm.

Principle of MLP

Concerning the proposed network, MLP network will be used which is a supervised learning technique that belongs to the Feed-Forward neural network family. It transfers information from an input dataset to a separate set of outputs in one direction [48]. An MLP is made up of nodes and layers that begin with the input layer and progress via one or more hidden levels to the output layer. Each node is connected to the next one by a weight and bias value (threshold), and each node is fully connected to the next layer. If the output of any node exceeds a certain threshold, that node is activated and the data is sent to the next layer of the network; otherwise, the data is not transferred. A neuron (node) in a neural network is a mathematical function that gathers and categorizes data based on a predefined schema. The network closely resembles statistical methods like curve fitting and regression analysis. The input weights are similar to the coefficients in a regression equation. However more complex initialization schemes can be used, they are frequently initialized to small random values. The weighted inputs are added and passed through an activation function known as a transfer function. An activation function is a straightforward mapping of summed weighted input to neuron output. It is referred to as an activation function because it controls the activation threshold of the neuron as well as the strength of the output signal. Non-linear activation functions have mostly been used. This permits the network to combine inputs in more complex ways, providing a richer capability in the functions it can model. Non-linear functions such as the logistic, also known as the sigmoid function (Equation 1 [49]), which produces a value between 0 and 1 with an s-shaped distribution, and the hyperbolic tangent function tanh (Equation 2 [49]), which produces the same distribution over the range -1 to +1.

There are several activation functions $\Phi(.) : R \rightarrow R$, the mostly used ones are:

a) Sigmoid function: $\Phi(x) = \frac{1}{1+e^{-x}}$ (1)

b) Hyperbolic tangent sigmoid (tanh) transfer function: $\Phi(x) = \frac{2}{\left(\left(1+e^{-x}\right)-1\right)}$ (2)

Different activation functions can be used in the same model.

The type of problem modeled strongly influences the activation function used in the output layer, e.g., A regression problem may have a single output neuron with no activation function. A single output neuron in a binary classification problem may use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for class 1. This can be converted into a crisp class value by using a threshold of 0.5 and snapping values less than the threshold to 0 and values greater than the threshold to 1. In the output layer of a multi-class classification problem, there may be multiple neurons, one for each class, e.g., three neurons for the three classes in the famous iris flowers classification problem. In this case, a softmax activation function can be used to determine the probability of the network correctly predicting each of the class values. A crisp class classification value can be obtained by selecting the output with the highest probability.

Hidden layers are those that follow the input layer and are not directly exposed to the input. The most basic network structure is a single neuron in the hidden layer that outputs the value directly. Hidden layers adjust the input weightings until the neural network’s margin of error is as small as possible. The final hidden layer is known as the output layer, and it oversees producing a value or vector of values that directly relate to the problem’s format.

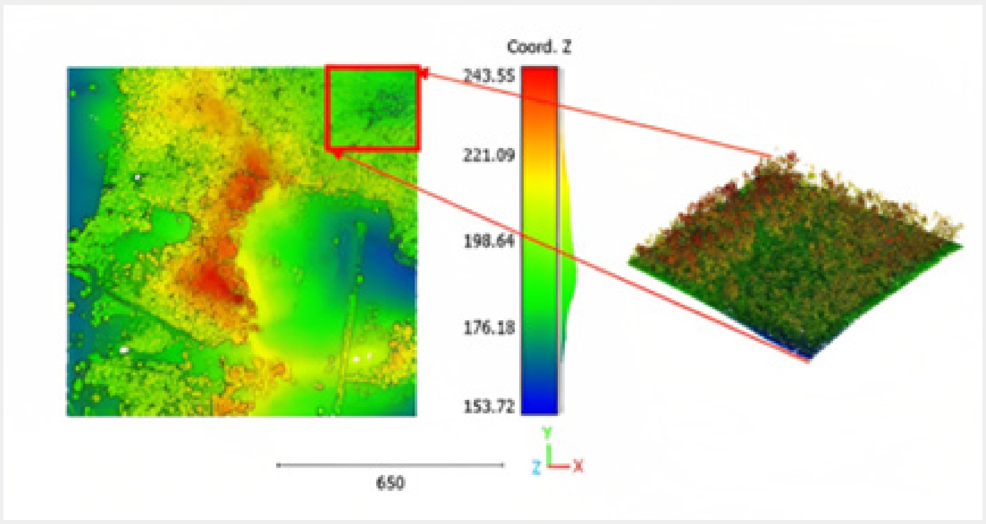

Figure 3: First and second point cloud of Lockyer Valley north 2015 (Table 1).

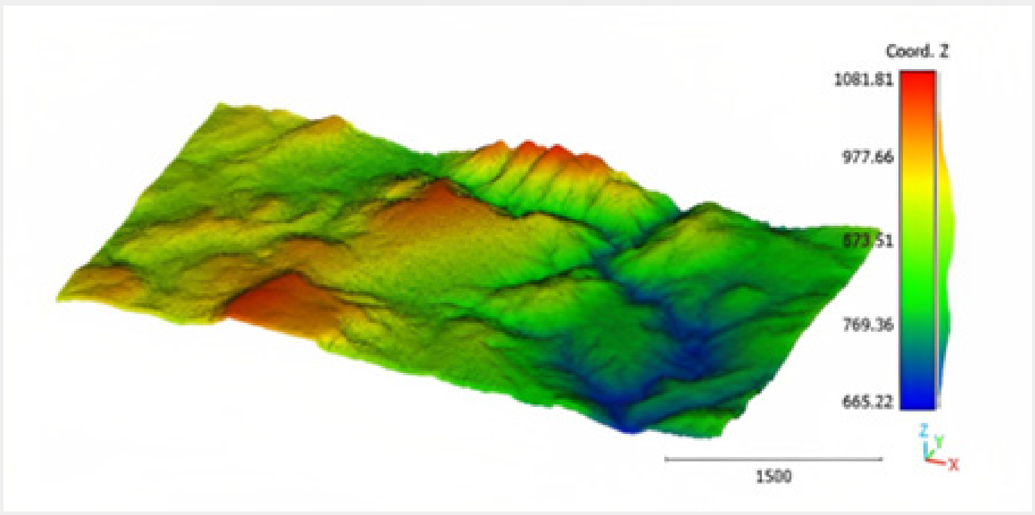

Figure 4: Gisborne, New Zealand 2018-2020 point cloud (Table 1).

How does MLP work

By training the data set, MLP learns a function $S(.): R^m \rightarrow R^o$, where m is the number of dimensions for input and o is the number of dimensions for output [50,51]. Between the input and output layers, one or more nonlinear hidden layers may exist. M neurons are found in the hidden layers. For each passage the weighted sum is calculated by a linear combination of input features $X = [x_1, x_2, x_3, \ldots, x_n]$ and their weights $W = [w_1, w_2, \ldots, w_n]$ plus a bias term b [52].

The weighted sum of the inputs and the bias were calculated using Equation 3.

$$S = b + \sum_{k=1}^{m} x_{ik} W_{ik} \qquad (3)$$

The activation function determines the behavior of the node [53]:

$$U = \Phi(S) = \Phi(b + \sum_{k=1}^{m} x_{ik} w_{ik}) = \begin{cases} 1 \ if \ S \geq 0 \\ 0 \ if \ S < 0 \end{cases} \quad (4)$$

Figure 5 shows an example of a neural network with two hidden layers, as well as the functions and parameters used in each step of the process starting from inputs to outputs.

Deep learning pipeline

A Deep Learning (DL) pipeline $h : D_1 \rightarrow D_2$ is a sequential combination of various algorithms that transforms a feature vector from the input dataset $x \in D_1$ into a target value $y \in D_2$ of the output dataset, a class label for a classification problem. The triplet (g, A, λ) defines a DL pipeline $P_{g,A,\lambda}$ with g ∈ G a valid pipeline structure, $A \in A^{'} = \{A^{(1)}, A^{(2)}, ..., A^{(n)}\}$ a vector consisting of the selected algorithm for each node and λ a vector comprising the hyperparameters of all selected algorithms.

By joining the probability distribution $P(D_1, D_2)$ of the feature space D_1 and target space D_2 known as a generative model, the pipeline trained on the generative model P noted as $P_{g,A,\lambda,P}$ [37].

The performance of the pipeline $h_{(g,A,\lambda)}$ given by a loss function $\varsigma(.,.)$ and a generative model $P(D_1, D_2)$ and calculated as

$$R\left(P_{g,A,\lambda,p}, P\right) = \mathrm{E}\left(\left(h(D_1), D_2\right)\right) = \int \varsigma\left(h\left(D_1\right), D_2\right) dP(D_1, D_2) \quad (5)$$

With $h(D_1)$ is the predicted output of $P_{g,A,\lambda,P}$.

Creating a DL pipeline for a specific ML assignment may be divided into three parts: first, establish the structure of the pipeline, such as how many preprocessing and feature engineering processes are required, how the data flows through the pipeline, and how many models must be trained. Then, for each stage, an algorithm must be chosen in the presented case MLPClassifier algorithm was chosen. Finally, the relevant hyperparameters for each algorithm must be chosen. To evaluate pipeline performance, all stages must be performed [37].

The Pipeline object can combine several transformers and an estimator to create a combined estimator to, e.g., apply dimension reduction before fitting [54]. The Pipeline is often used in combination with FeatureUnion that concatenates the output of transformers into a composite feature space. In contrast, Pipelines only transform the observed data [39].

Pipeline serves multiple purposes:

a) Convenience and encapsulation: only one call to fit and predict the data is required to fit a whole sequence of estimators.

b) Joint parameter selection: the ability to perform a grid search over the parameters of all estimators in the pipeline at the same time.

c) Safety: By ensuring that the same samples are used to train the transformers and predictors, pipelines help to avoid statistics from the test data leaking into the trained model in cross-validation.

Suggested workflow

Figure 6 illustrates the general workflow of using the MLP pipeline algorithm. The process starts with splitting the data in order to test the MLP pipeline algorithm that was created, trained, and validated. Finally, use it to make predictions.

Once the labeling phase of the preparation of the data ends using CloudCompare software mentioned in section 3.1, the second phase of the workflow can be summarized in Figure 7 as follows:

a) Split the labeled dataset into 70% for the Train set, 10% for the validation set, and 20% for the test sets (A).

b) Develop the algorithm of the MLP pipeline using python, NumPy, and Sklearn libraries (B).

c) Train the model with different parameters using the train set (C). To get the best parameters a function called GridSearchCV() was used from the Sklearn library [54] that can be used to determine optimal values for a model's hyperparameters. Choosing the best hyperparameters has a big impact on the performance of the approach.

d) Test the trained and the validated model using the test set to confirm the efficacity of its performance (E).

The output of this model is a predicted label for each point (1 for vegetation and 0 for terrain). The final outcome was represented in the results section (Figures 28-31).

This workflow starts by splitting the dataset into a training set and a testing set, using a function called -train_test_split- from sklearn library [54].

It takes as parameters the grouped features of the point clouds, the labels (1 or 0), the size of the test set for in this case 0.2 (e.g., 20%) of the original data, and the last parameter is random state which controls the shuffling process. This function returns four variables; the first contains the training set, the second contains the labels of the training set, the third is the test set, and the last one is the labels of the test. It separates the labels of each set (training and testing) in a separate variable, so that the labels of the test set will be considered as a reference model to compare it with the obtained labels by the proposed approach in order to get the accuracy of the approach.

Next, the pipeline was created using the neural network MLPClassifier algorithms from scikit-learn library [54] and the StandardScaler function that removes the mean and scales each feature/variable to unit variance. The parameters of the pipeline used [54] are: steps which is a list of Estimator objects, memory is string or object, the default value is none used to cache the fitted transformers of the pipeline, verbose is a Boolean its default value is false, if its value is true, the elapsed time while fitting each step will be printed as it is accomplished. Then, the MLP pipeline algorithm is trained and validated, after varying and optimizing different hyperparameters of the algorithm mentioned in Section 4.2 as the input layer elements, the number of Hidden Layers (HL), the activation functions, and the alpha value used. The process behind this step is represented in Figure 7. Finally, test the algorithm and analyze the score obtained if it will allow using the MLP pipeline algorithm to get predictions.

The accuracy score was calculated to evaluate the performance of the approach, using the function described in Equation 6 [54].

$$Accuracy\left(\hat{y}_l, y_i\right) = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predicitons} = \frac{1}{m}\sum_{i=0}^{m-1} 1\left(\hat{y}_l = y_i\right) \quad (6)$$

Where $\hat{y}_l$ is the predicted value of the i-th sample and y_i is the corresponding true value, m is the number of input dimensions.

The best accuracy results were obtained with 10 HL, alpha=0.0001, solver=" adam", activation=" tanh". Finally, after the MLP pipeline was trained and validated, it was tested with a test set to estimate the accuracy.

The difference between using an MLP algorithm and an MPL pipeline algorithm is that a machine learning pipeline is a series of steps used to create, deploy, and monitor a machine learning model. The method is used to map a machine learning model's entire development, training, deployment, and monitoring. It is frequently employed to automate the classification process.

The process of defining each step as a unique module of the overall process is an important aspect of building machine learning pipelines. This modular approach assists organizations in viewing machine learning models holistically, assisting in the organization and management of the end-to-end process. However, it also provides a solid foundation for model scaling, as individual modules within the machine learning pipeline can be upscaled or downscaled.

Feature Calculation

A mean plane as well as a normal vector can be fitted through one point and its neighborhood using the least square theory [15].

To determine the neighborhood of each LiDAR point, the neighborhood matrix (N_m) is defined. This matrix consists of n rows (n is the number of points in the point cloud list) where row number i contains the neighboring point numbers of point number I [16]. After calculating N_m, the considered features, along with the normal vectors of the point cloud, can be calculated using the eigenvectors of the covariance matrix. The features of the point cloud are calculated based on different combinations of eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and eigenvectors of Cov[P, P] of a point P [55] (Equation 7).

$$Cov(A) = \frac{1}{|A|}\sum_{p\in A}(p-\bar{p})(p-\bar{p})^T \qquad (7)$$

Where p is a given point and $\bar{p}$ is the centroid of the neighborhood A.

Assuming that λ_i is the eigenvalues $_{i\in\{1,2,3\}}$, the main geometric features based on the eigenvalues and eigenvector are: the sum of eigenvalues, omnivariance, eigenentropy, anisotropy, planarity, linearity, PCA1, PCA2, surface variation, Sphericity, verticality, anisotropy, linearity, PCA1 and, PCA2 [55] (see Table 2). The equations of these features are mentioned in Table 2.

Table 2: Definition of geometric features.

Feature	Definition	Equation Number
Sum of eigenvalues	$\sum \lambda_i = \lambda_1 + \lambda_2 + \lambda_3$	8
Ominvariance	$O = \sqrt[3]{\lambda_1 \times \lambda_2 \times \lambda_3}$	9

Eigenentropy	$E = -\sum \lambda_i \ln(\lambda_i)$	10
Planarity	$\mathrm{P} = \frac{\lambda_2 - \lambda_3}{\lambda_1}$	11
Sphericity	$S = \frac{\lambda_3}{\lambda_1}$	12
Surface variation	$Sv = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$	13
Verticality	$V = 1 - \lvert n_z \rvert$	14
Anisotrop	$A = \frac{\lambda_1 - \lambda_3}{\lambda_1}$	15
Linearity	$L = \frac{\lambda_1 - \lambda_2}{\lambda_1}$	16

PCA1	$PCA1 = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$	17
PCA2	$PCA2 = \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda_3}$	18

The verticality is derived from the vertical component n_z of the normal vector $n \in \mathbb{R}^3$ [56]. λ_i are eigenvalues.

Geometric Feature Analysis

After being presented in the last section, the equations of the main features employed in the literature, which can be calculated from one point and its neighborhood [57], and the more effective features that highlight the difference between the terrain and the vegetation can be selected. For this purpose, two datasets of different topography areas are used. The first dataset is of a flat area which was measured by plane (first dataset in Table 1), and the second dataset is of a mountain area which was measured by Unmanned Aerial Vehicle (UAV) (drone) (third dataset in Table 1).

To analyze the different geometric features, the two employed datasets are classified manually (point per point) into two classes: terrain and vegetation. At this stage, it is important to mention that Shan & Toth [2] suppose that manual classification is not only more accurate than automatic classification, but it can produce classes that can be used as references to estimate the accuracy of the automatic classification. Thereafter, each geometric feature is calculated for the two separated classes for the two used point clouds. In order to scrutinize the obtained feature values, a histogram graph is calculated for the list of feature values of each class in the given point clouds. The tested feature list is the sum of eigenvalues, ominvariance, eigenentropy, planarity, sphericity, surface variation, Anisotrop, Linearity, PCA1, PCA2, and verticality.

Figures 8a, 8b, 9a & 9b illustrate the histograms of the values obtained from the calculated eigenentropy features. It can be noted that the value distributions are strongly distinctive between the vegetation and terrain classes. The value difference and distribution become more representative in the mountain area, where most of the terrain points have negative values in contrast to the vegetation points. Therefore, this feature can be useful for recognizing the vegetation class from the terrain class.

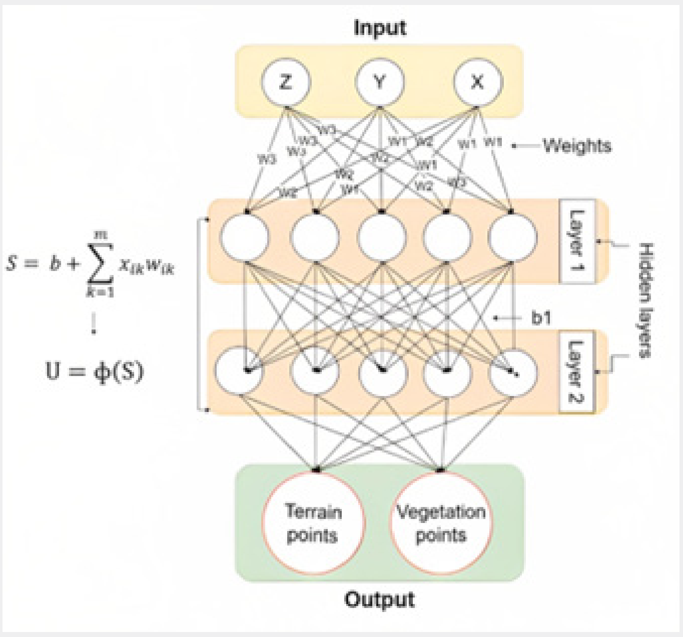

Figure 5: Example of neural network with 2 hidden layers.

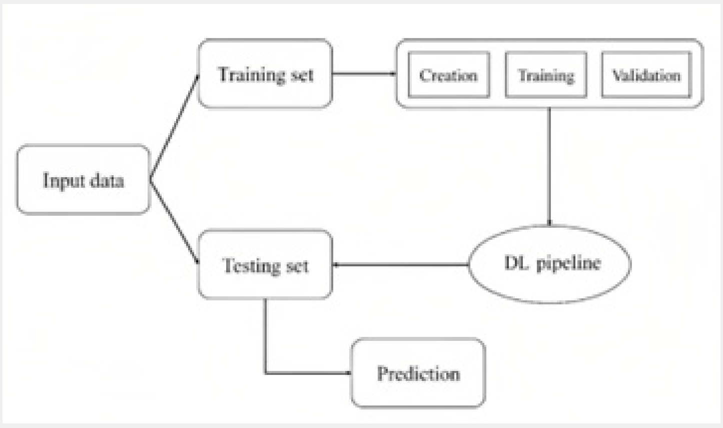

Figure 6: DL pipeline general workflow.

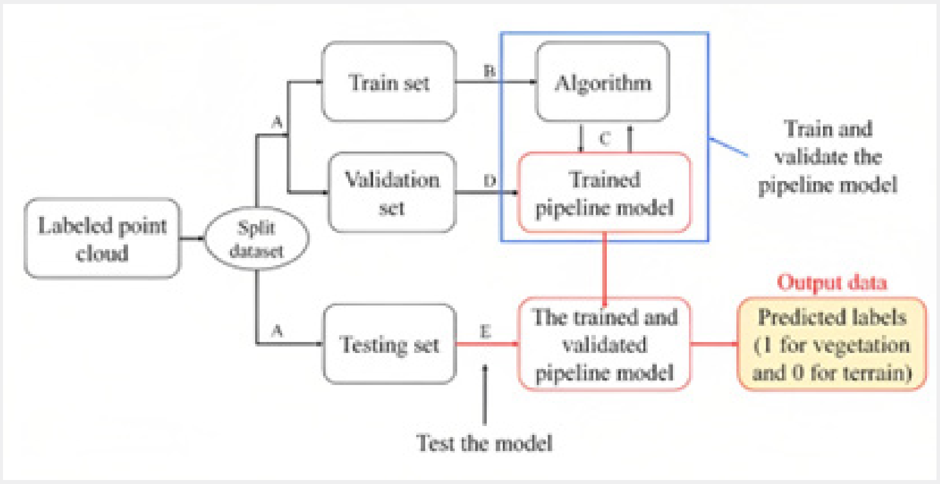
Train set
B
Algorithm
A
C
Train and validate the pipeline model
Validation set
D
Trained pipeline model
Labeled point cloud
Split dataset
Output data
A
Testing set
E
The trained and validated pipeline model
Predicted labels (1 for vegetation and 0 for terrain)
Test the model

Figure 7: The phase of creating, evaluating, and testing the MLP pipeline algorithm (model).

Figures 10a, 10b, 11a & 11b illustrate the histograms of the values obtained of the calculated ominvariance features. The value distributions are obviously distinct between the vegetation and terrain classes. The estimation and distribution become more indicative in the mountain area, where the majority of terrain points have values between 1 and 2 compared to vegetation points that have the majority of values between 0.1 and 0.5. As a result, this feature can help distinguish the vegetation class from the terrain class.

Figures 12a, 12b, 13a & 13b illustrate the histograms of the values obtained from the calculated planarity features. The figures in both areas show that every histogram has its slightly distinct distribution of points but unfortunately, they have similar distribution intervals between 0 and 0.8. This observation weakens the importance of this feature for the recognition of vegetation from the terrain. In fact, this conclusion is sufficiently logical because this feature expresses the presence of planner surfaces which is not valid in vegetation and terrain classes.

Figures 14a, 14b, 15a & 15b illustrate the histograms of the values obtained from the calculated sphericity features. in the flat area, the feature values have different distribution intervals where the vegetation interval is [0, 0.4] whereas the terrain interval is [0, 0.01]. Despite this distribution in the case of mountain areas becoming less significant, the huge histogram form difference between vegetation and terrain reinforces the importance of this feature to separate terrain from vegetation.

Figures 16a, 16b, 17a & 17b illustrate the histograms of the values obtained from the calculated sum of eigenvalues features. It characterizes the difference between mountain as well as flat areas. Indeed, in the mountain area, the values less than 2 represent the vegetation class whereas the values greater than 2 represent the terrain class. Moreover, in the flat area, the distribution intervals as well as the histogram forms are distinctive between the two studied classes. Hence, the sphericity feature can play an important role in localizing the terrain and the vegetation points.

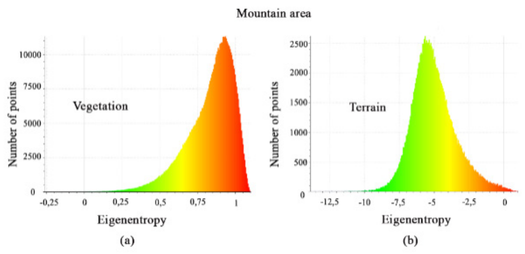

Figure 8: Histograms of eigenentropy feature values in mountain area. (a) Vegetation class; (b) Terrain class.

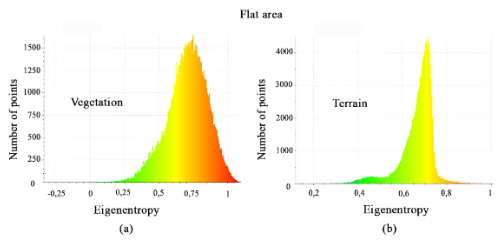

Figure 9: Histograms of eigenentropy feature values in flat area. (a) Vegetation class; (b) Terrain class.

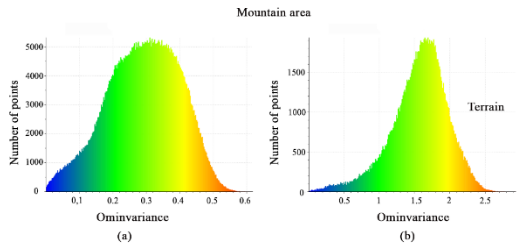

Figure 10: Histograms of ominvariance feature values in mountain area. (a) Vegetation class; (b) Terrain class.

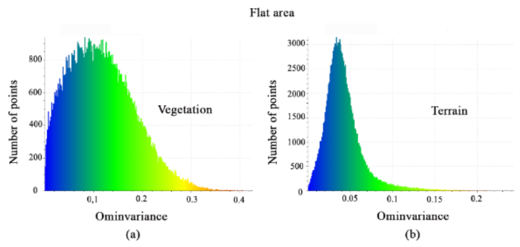

Figure 11: Histograms of ominvariance feature values in flat area. (a) Vegetation class; (b) Terrain class.

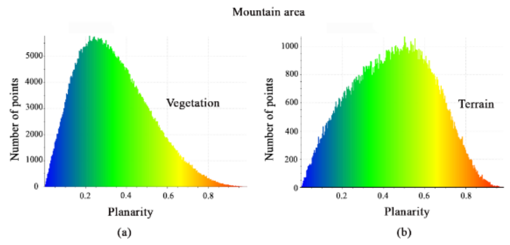

Figure 12: Histograms of planarity feature values in mountain area. (a) Vegetation class; (b) Terrain class.

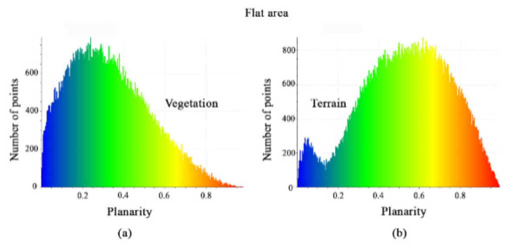

Figure 13: Histograms of planarity feature values in flat area. (a) Vegetation class; (b) Terrain class

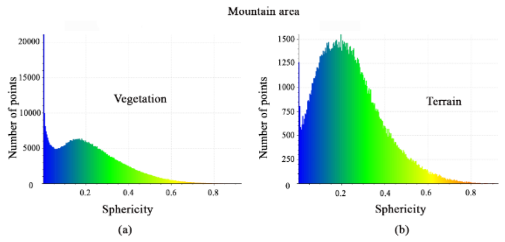

Figure 14: Histograms of sphericity feature values in mountain area. (a) Vegetation class; (b) Terrain class.

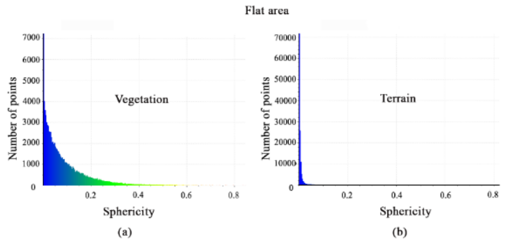

Figure 15: Histograms of sphericity feature values in flat area. (a) Vegetation class; (b) Terrain class.

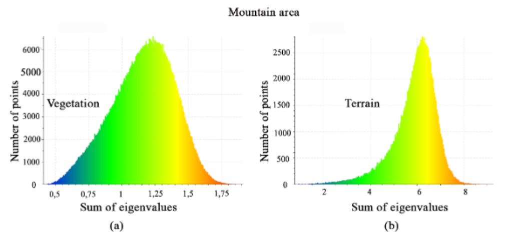

Figure 16: Histograms of sum of eigenvalues feature values in mountain area. (a) Vegetation class; (b) Terrain class

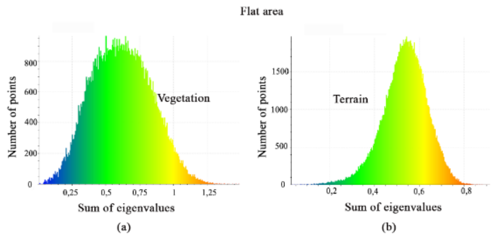

Figure 17: Histograms of sum of eigenvalues feature values in flat area. (a) Vegetation class; (b) Terrain class.

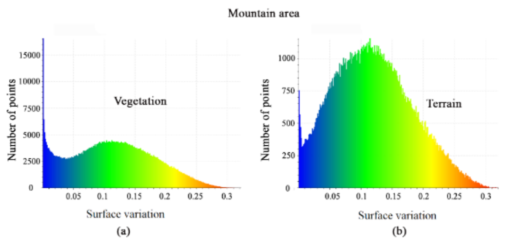

Figure 18: Histograms of surface variation feature values in mountain area. (a) Vegetation class; (b) Terrain class.

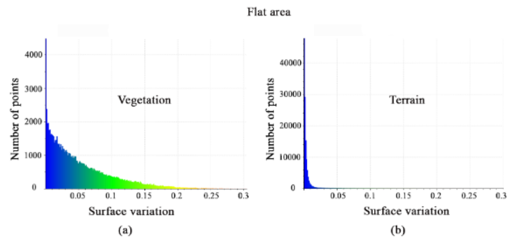

Figure 19: Histograms of surface variation feature values in flat area. (a) Vegetation class; (b) Terrain class.

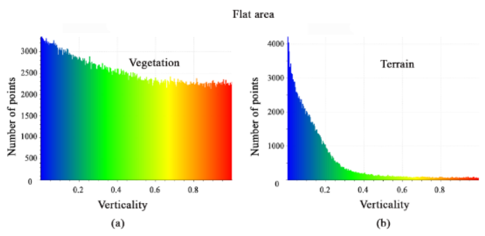

Figure 20: Histograms of verticality feature values in mountain area. (a) Vegetation class; (b) Terrain class

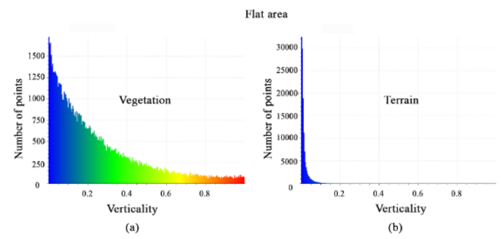

Figure 21: Histograms of verticality feature values in flat area. (a) Vegetation class; (b) Terrain class

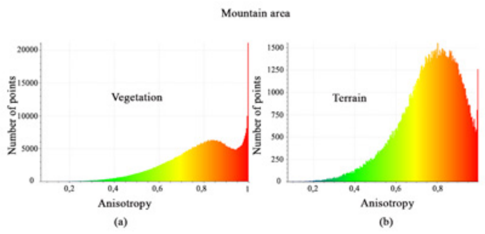

Figure 22: Histograms of anisotropy feature values in mountain area. (a) Vegetation class; (b) Terrain class.

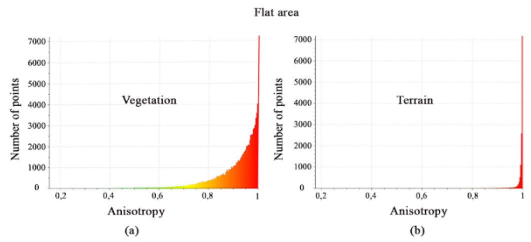

Figure 23: Histograms of anisotropy feature values in flat area. (a) Vegetation class; (b) Terrain class.

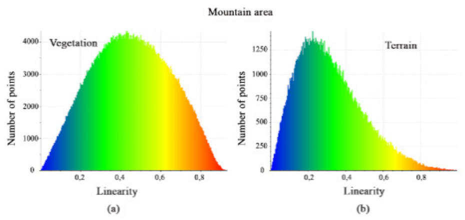

Figure 24: Histograms of linearity feature values in flat area. (a) Vegetation class; (b) Terrain class.

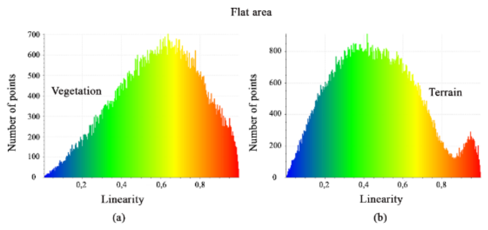

Figure 25: Histograms of linearity feature values in flat area. (a) Vegetation class; (b) Terrain class.

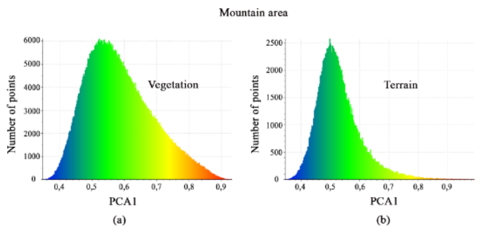

Figure 26: Histograms of PCA1 feature values in mountain area. (a) Vegetation class; (b) Terrain class.

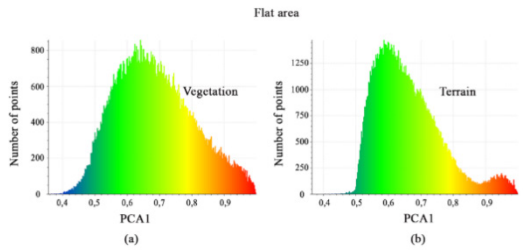

Figure 27: Histograms of PCA1 feature values in flat area. (a) Vegetation class; (b) Terrain class.

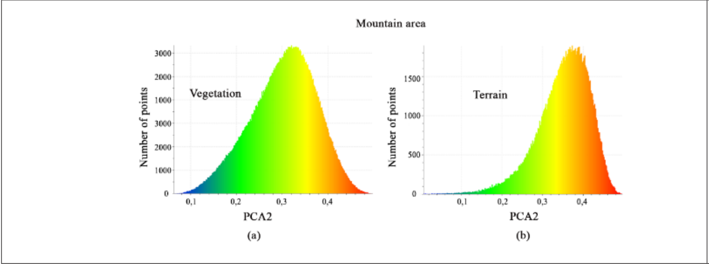

Figure 28: Histograms of PCA2 feature values in mountain area. (a) Vegetation class; (b) Terrain class.

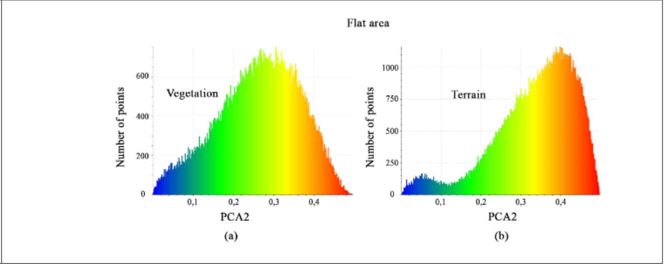

Figure 29: Histograms of PCA2 feature values in flat area. (a) Vegetation class; (b) Terrain class.

Figures 18a, 18b, 19a & 19b illustrate the histograms of the values obtained from the calculated surface variation feature. In mountain areas, the histograms show the distribution of points in the same interval though a great difference between the histogram forms. Unlike in the flat area, most terrain points have values equal almost to 0 in contrast to the vegetation class points which have values localized between 0 and 0.15. In conclusion, the surface variation feature can help enormously to carry out the classification of vegetation/terrain.

Figures 20a, 20b, 21a & 21b show histograms of the calculated verticality feature values. The histograms of mountain areas show that the point distribution has the same interval despite the considerable histogram form difference. Whereas in flat areas, in addition to the huge histogram form difference, most terrain points are located by the 0 value in opposite to vegetation class points decline quickly from 0 to 1. However, the great histogram distinguished behavior between terrain and vegetation in both flat and terrain areas making the verticality feature considered an efficient one.

Figures 22a, 22b, 23a & 23b show histograms of the calculated anisotropy feature values. In mountain areas, the histograms show that the distributions of points are in the same interval with different histogram forms, while in flat areas, the main of the terrain points have values equal almost to 1 and the vegetation class has values between 0.7 and 1. This considerable difference between histogram behaviors raises the importance of the anisotropy feature.

The calculated linearity feature values are shown in Figures 24a, 24b, 25a & 25b. The histograms in both areas show that the distribution of points is almost in the same interval. Moreover, despite the histogram spike in variant points, the difference between them is still insignificant and consequently, the linearity feature will not help in the two-goal classes separation. At this stage, it is important to note that this conclusion may be considered coherent because in the two target classes it is not anticipated to meet point gropes having linear relationships.

Figures 26a, 26b, 27a & 27b show the calculated PCA1 feature values. Both areas have the same interval of point distribution with similar histogram forms. These slight differences between the histogram of the PCA1 feature will not be able to help to separate the vegetation and terrain classes, which is why this geometric feature will not be considered.

The same last histogram analysis can be carried out on the PCA2 feature. Hence, the calculated PCA2 values that are shown in Figures 28a, 28b, 29a & 29b, illustrate that the distribution of points in both areas will not be benefited to distinguish between the two target classes.

To summarize the obtained results illustrated in Figures 8 to 29, a new factor is defined named Histogram Form Resemblance Mark (HFRM). This factor reflects the resemblance level between the two histograms. It considers the interval of point distribution (the horizontal axis values) as well as the histogram form.

$$HFRM = OP \times R \qquad (8)$$

Where OP is the overlap percentage between the two intervals of point distribution, and R is the histogram shape similarity mark. The R value can be estimated by the user where a value can be estimated between 1 when the two histogram shapes are exactly similar, and 0.5 when the two histogram shapes are completely different.

These factor values are between 0 where there is no resemblance between two histograms and 1 when the two histograms are exactly similar (Table 3). In Table 3, for each feature six values are estimated. First, the Vegetation Class in the Mountain area (VCM) represents the interval of point distribution of the vegetation histogram in the mountain area (the horizontal axis values). Second, the Terrain Class in the Mountain area (TCM) represents the interval of point distribution of the terrain class histogram in the mountain area. Third, the Vegetation Class in a flat area (VCF) represents the interval of point distribution of the vegetation class histogram in the flat area. Fourth, the Terrain Class in the Flat area (TCF) represents the interval of point distribution of the terrain class histogram in the flat area. Finally, the HFRM factor is calculated two times to compare the case of the mountain and flat areas. The basic adopted hypothesis says if the similarity level (HFRM) between vegetation and terrain feature histograms in the same point cloud is high, hence the employment of this feature in the input layer of the DL algorithm will not be useful and vice-versa. To select the features that will be used in the DL algorithm, it is sufficient to select features having HFRM values smaller than 0.5 in Table 3. Consequently, the benefit features for the ML algorithm are the sum of eigenvalues, ominvariance, eigenentropy, Anisotropy, sphericity, surface variation, and verticality. In the same context, four geometric features are rejected because their HFRM values are greater than 5 which are Planarity Linearity PCA1 and PCA2.

Table 3: Summarizing of feature values illustrated in Figures 7 to 28. HFRM is Histogram Form Resemblance Mark; VCM: Vegetation Class in Mountain area; TCM: Terrain Class in Mountain area; VCF: Vegetation Class in Flat area; TCF: Terrain Class in Flat area.

Feature	VCM	TCM	HFRM	VCF	TCF	HFRM
Sum of eigenvalues	0.75 – 1.5	4 – 7.5	0	0.25 - 1	0.4 – 0.7	0.5
Omnivariance	0.1 – 0.5	1 – 2.1	0.2	0 – 0.26	0 – 0.1	0.4
Eigenentropy	0.5 – 1.1	-7.5 – -2.5	0	0.5 - 1	0.6- 0.75	0.5
Planarity	0.5 – 0.6	0.2 – 0.8	0.8	0 – 0.7	0.2 – 0.9	0.8
Sphericity	0 – 0.4	0 – 0.5	0.5	0 – 0.2	0 – 0.025	0.3
Surface variation	0 – 0.25	0 – 0.25	0.5	0 – 0.1	0 – 0.01	0.2
Verticality	0 - 1	0 - 1	0.5	0 - 1	0 – 0.05	0.2
Anisotropy	0.3-1	0.2-1	0.5	0.6-1	0.995-1	0.2
Linearity	0-1	0-1	0.7	0-1	0-1	0.8
PCA1	0.4-0.92	0.37-0.9	0.8	0.4-1	0.45-1	0.8
PCA2	0.05-0.5	0.1-0.5	0.75	0-0.5	0.1-0.52	0.8

At this stage, it is important to mention that one feature can be useful to recognize one object class in the point cloud and may be useless to recognize another object class in the point cloud. The obtained feature list will be used to obtain the neighborhood characteristics, and the shape to understand the data distribution. In other words, this result will help the ML algorithm to recognize and differentiate the vegetation and the terrain points in both mountain and flat areas. The given features used for purely geometric classification problem: to learn the suggested ML model to classify the points cloud as a binary classification into two classes vegetation (label = 1) and terrain (label = 0).

Results and Discussion

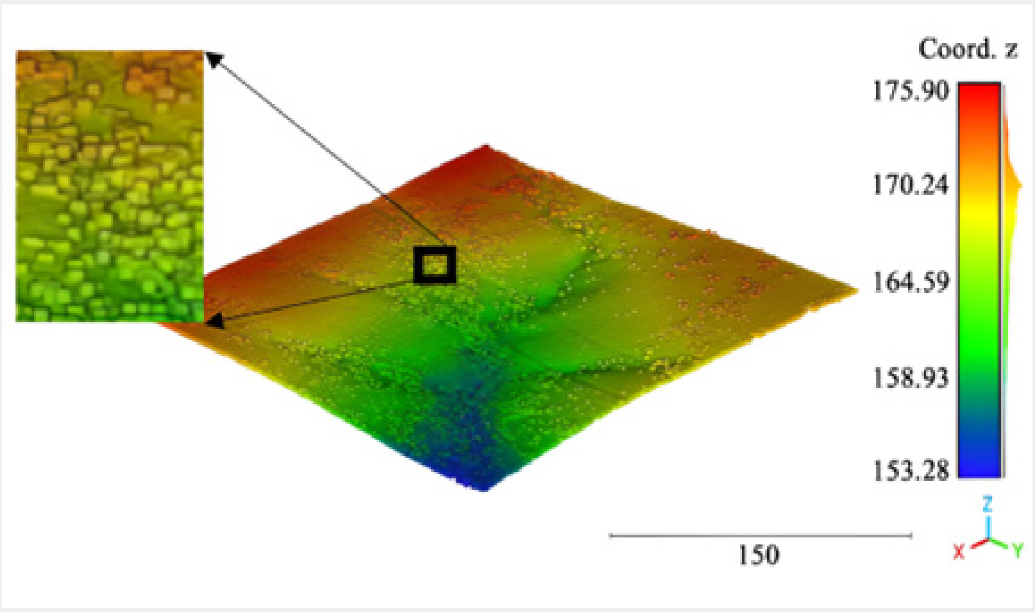

Figure 30: Extracted terrain points from flat area point cloud.

The suggested approach classifies the point cloud into two classes: vegetation and terrain, using the adapted MLP classifier ML pipeline. The suggested approach was tested using four point-clouds, the first two ones is of a flat area (Figure 1 & 3), and all points of the third cloud are in mountain area (see Figure 2 & Table 1). Figures 30 & 31 represent the obtained results of applying the suggested classification algorithm on the flat area point cloud whereas Figures 32 & 33 illustrate the results obtained in the mountain area point cloud. These figures show that the vegetation/ terrain classification is carried out accurately in both cases. In fact, the developed approach provides an accuracy score calculated by Equation 6 equal to 99.03% in the flat area, and a score equal to 99.87% in the mountain area. The score is calculated by comparing the algorithm's predicted results with manual classification of the point cloud as a reference model. Indeed, the point cloud quality (point density and accuracy) of the mountain area is much greater than the quality of the flat area point cloud. That is why the estimated classification accuracy in the mountain cloud was higher. In this context, this accuracy is calculated using a manually classified point cloud as a reference model [58]. Nevertheless, the classification accuracy in the two cases was excellent (greater than 99%), which confirms the high efficiency of the suggested approach.

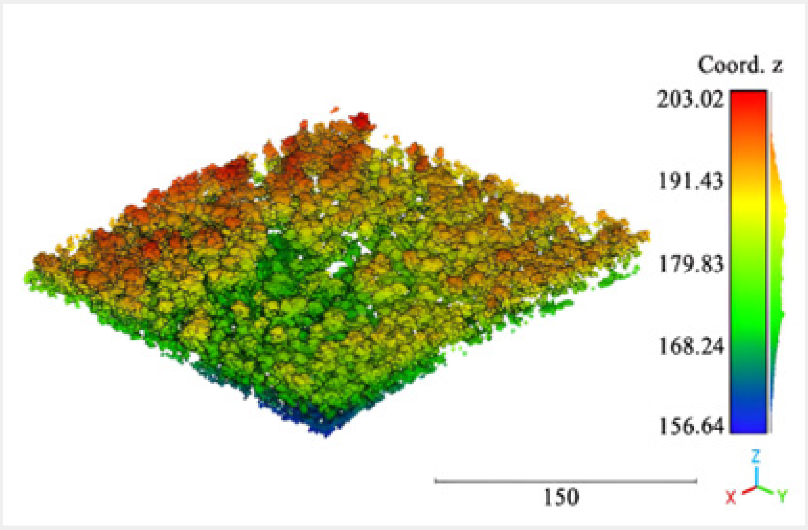

Figure 31: Extracted vegetation points from flat area point cloud.

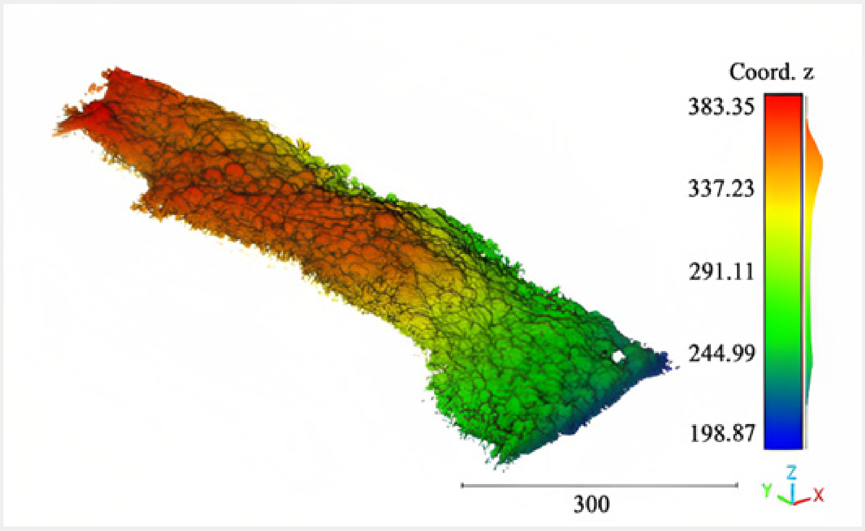

Figure 32: Extracted vegetation points in mountain area point cloud.

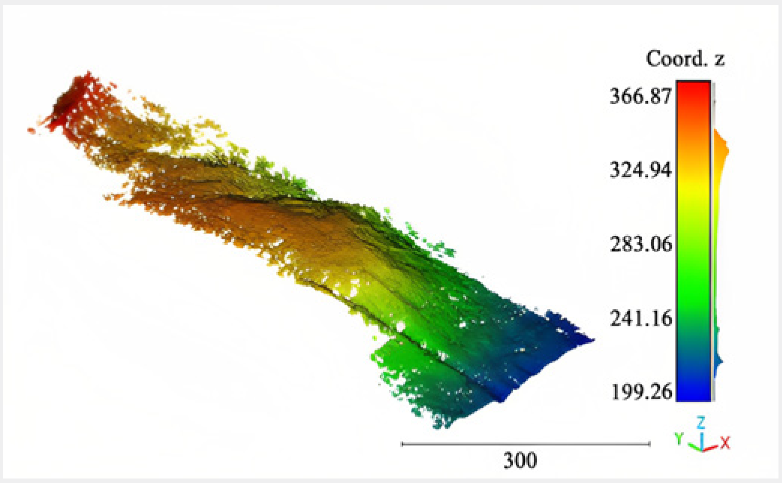

Figure 33: Extracted terrain points in mountain area point cloud.

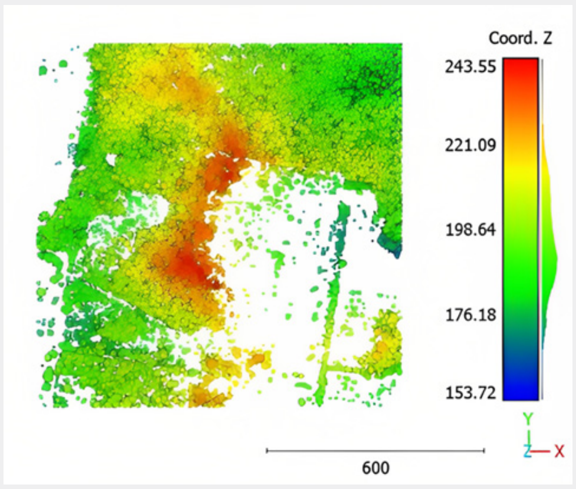

Figure 34: Extracted vegetation points of the second point cloud (Table1).

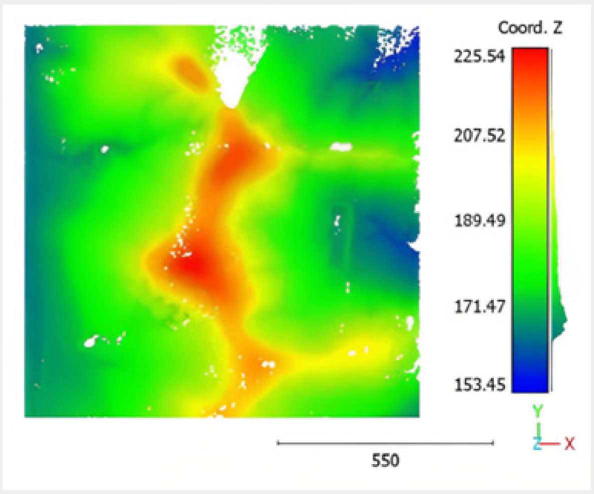

Figure 35: Extracted terrain points of the second point cloud (Table1).

At this stage, it is unavoidable to note that the aboveground parts of one tree are the crown and the trunk. Figures 30-33 show that the tree crowns are perfectly recognized. Moreover, the upper 90% of tree trunks are also perfectly established. Unfortunately, the trunk linkage parts between trees and ground are sometimes considered as ground (see the zoomed area in Figure 30). Indeed, the LiDAR point density and distribution of these trunk parts can disturb the classification algorithm and generate this miss-classification. Moreover, in the data labeling stage before the training step, the accuracy of point labeling of the linkage trunk parts may be low for the same last reasons. Nevertheless, more investigation is required to solve this issue.

To effectively test the algorithm and make sure of the obtained accuracy value, it is applied on larger areas of datasets 2 and 4 (see Figure 3 & 4 in addition to Table 1). Indeed, the topography of the newly tested areas is a mix of mountain and flat. The obtained results are illustrated in Figures 34-37 where the obtained accuracy score is equal to 98.03 % and 98.12 % consecutively. This result on large-scale data and complex ground topography confirms and reinforces the efficiency of the suggested classification approach.

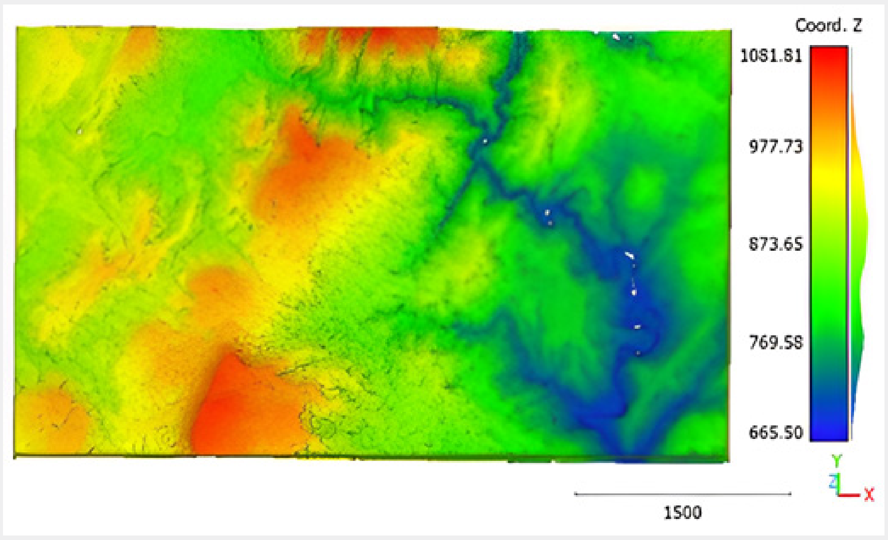

Figure 36: Extracted vegetation points of Gisborne (Table1).

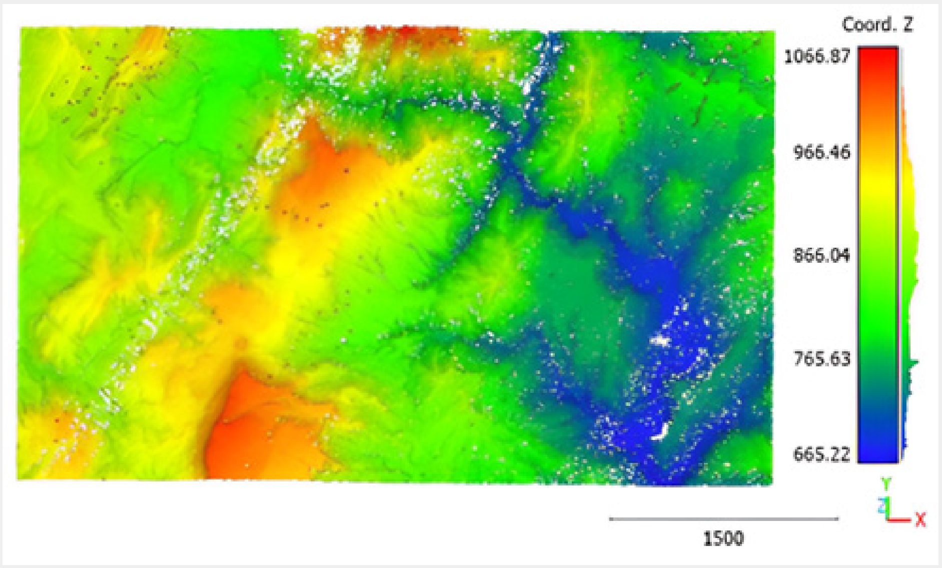

Figure 37: Extracted terrain points of Gisborne (Table1).

At this stage, in the context of the efficacity evaluation of the suggested approach, it is beneficial to compare the accuracy obtained by the suggested algorithm with the accuracies of similar suggested classification algorithms in the literature. Hence, Vega et al. [59] introduce the rule-based PTrees algorithm that represents a multi-scale dynamic point cloud segmentation method for extracting forest trees from LiDAR point clouds using raw elevation values (Z) and computing height (H = Z ground elevation). This approach has been tested in three different forest types, detecting 82% of the trees with a false detection rate of less than 10%. Kwak et al. [60] present a method for detecting individual trees and estimating tree heights in three South Korean forests using LiDAR data. To detect treetops from a Digital Canopy Model (DCM) using the extended maxima transformation of morphological image-analysis methods. To classify individual tree species from hyperspectral data with a high spatial and spectral resolution, Mäyrä et al. [61] compare the performance of three-dimensional convolutional neural networks (3D-CNNs) with the support vector machine, random forest, gradient boosting machine, and artificial neural network. Luo et al. [35] propose a deep learning framework for detecting trees in complex forests using UAV laser scanning point clouds that is based on a designed multi-channel information complementarity representation. The proposed method is divided into two parts: ground filtering and tree detection. The proposed framework achieves a mean recall of 89.23% and an F1-score of 87.04%.

Table 4 represents the accuracy comparison between the suggested approach and similar other approaches in the literature. From Table 4, it can be noted that the accuracy score obtained by the suggested classification approach is the highest compared to the earliest studies of either rule-based or ML algorithms, which confirms the efficacity of the suggested approach.

Table 4: Accuracy comparison of suggested approach with similar other approaches in literature.

Approaches	Approach Family	Accuracy Score (%)
Vega et al. [59]	Rule-based	82
Kwak et al. [60]	Rule-based	86.7
Luo et al. [35]	Rule-based	89.23
Mäyrä et al. [61]	ML (3D-CNNs)	87
Mäyrä et al. [61]	ML (Support vector machine)	82.4
Mäyrä et al. [61]	ML (neural network)	81.7
Suggested approach	ML (Deep learning Pipeline)	98.03

Conclusion

This paper adapted the deep learning pipeline algorithm based on the MLP Neural Network to automatically classify the forest LiDAR point cloud. For this purpose, the parameters of the ML algorithm are adapted to obtain the highest possible performance through optimization of the algorithm parameters such as input layer elements, number of hidden layers, used activation functions, and alpha value. Moreover, and in the same context, the geometric features of LiDAR points were analyzed to recognize the effective features that will be employed in addition to the point coordinates within the classification algorithm input layer. In this context, eleven features are tested, and then four of them were excluded. To evaluate the efficiency of the suggested classification approach, two LiDAR point clouds were tested, one of a flat area and the other of a mountain area with different point densities and accuracies. The obtained high classification accuracy when it was applied on a large scale and complex topography areas (almost 98.03 %) confirmed the great effeminacy of the suggested algorithm.

In future work, it is unavoidable to test the proposed classification network on a large number of LiDAR datasets with different topography, vegetation kinds, and LiDAR data quality before adopting the final version of the suggested algorithm. Also, the issue of lower trunk part labelling and recognition needs improvement. Furthermore, more investigations are required to extend the suggested algorithm to classify more complicated scene point clouds such as urban areas where the point cloud may represent a long list of classes such as terrain, vegetation, buildings, roads, powerlines, and railways. Finally, the fact of obtaining high-quality results using a supervised ML algorithm must not distract the concentration from the unsupervised machine learning techniques that do not need the data training step.

Acknowledgment

Thanks to Paul Reed Managing Director of East Coast Surveys (Aust) Pty Ltd and CloudXPlus Pty Ltd companies for providing the LiDAR dataset of Figure 2 which was captured in Queensland, Australia, http://www.eastcoastsurveys.com.au (accessed on 19 November 2022).

References

1. Tarsha Kurdi F, Gharineiat Z, Campbell G, Dey EK, Awrangjeb M (2021) Full Series Algorithm of Automatic Building Extraction and Modelling from LiDAR Data. In Proceedings of the 2021 Digital Image Computing: Techniques and Applications (DICTA); IEEE: Gold Coast, Australia, pp.

01–08.

2. (2018) Topographic Laser Ranging and Scanning: Principles and Processing. In: Shan J, Toth CK (Eds.), CRC Press (2nd edn): Second edition. | Boca Raton: Taylor & Francis, CRC Press, ISBN 978-1-315-15438-1.

3. Yang Q, Su Y, Hu T, Jin S, Liu X, et al. (2022) Allometry-Based Estimation of Forest Aboveground Biomass Combining LiDAR Canopy Height Attributes and Optical Spectral Indexes. Forest Ecosystems 9: 100059.

4. Loh HY, James D, Ioki K, Wong WVC, Tsuyuki S, et al. (2022) Estimating Aboveground Biomass Changes in a Human-Modified Tropical Montane Forest of Borneo Using Multi-Temporal Airborne LiDAR Data. Remote Sensing Applications: Society and Environment 28: 100821.

5. Jiang F, Sun H, Ma K, Fu L, Tang J (2022) Improving Aboveground Biomass Estimation of Natural Forests on the Tibetan Plateau Using Spaceborne LiDAR and Machine Learning Algorithms. Ecological Indicators 143: 109365.

6. Chen M, Qiu X, Zeng W, Peng D (2022) Combining Sample Plot Stratification and Machine Learning Algorithms to Improve Forest Aboveground Carbon Density Estimation in Northeast China Using Airborne LiDAR Data. Remote Sensing 14(6): 1477.

7. Fekry R, Yao W, Cao L, Shen, X (2022) Ground-Based/UAV-LiDAR Data Fusion for Quantitative Structure Modeling and Tree Parameter Retrieval in Subtropical Planted Forest. Forest Ecosystems 9: 100065.

8. Xiong L, Lagomasino D, Charles SP, Castañeda-Moya E, Cook BD, et al. (2022) Quantifying Mangrove Canopy Regrowth and Recovery after Hurricane Irma with Large-Scale Repeat Airborne Lidar in the Florida Everglades. International Journal of Applied Earth Observation and Geoinformation 114: 103031.

9. Ma H, Zhou W, Zhang L (2018) DEM Refinement by Low Vegetation Removal Based on the Combination of Full Waveform Data and Progressive TIN Densification. ISPRS Journal of Photogrammetry and Remote Sensing 146: 260-271.

10. Nie S, Wang C, Dong P, Xi X, Luo S, et al. (2017) A Revised Progressive TIN Densification for Filtering Airborne LiDAR Data. Measurement 104: 70-77.

11. Zhang J, Lin X (2013) Filtering Airborne LiDAR Data by Embedding Smoothness-Constrained Segmentation in Progressive TIN Densification. ISPRS Journal of Photogrammetry and Remote Sensing 81: 44-59.

12. Roussel JR, Auty D, Coops NC, Tompalski P, Goodbody TRH, et al. (2020) LidR: An R Package for Analysis of Airborne Laser Scanning (ALS) Data. Remote Sensing of Environment 251: 112061.

13. Yu D, He L, Ye F, Jiang L, Zhang C, et al. (2022) Unsupervised Ground Filtering of Airborne-Based 3D Meshes Using a Robust Cloth Simulation. International Journal of Applied Earth Observation and Geoinformation 111: 102830.

14. Fareed N, Flores JP, Das AK (2023) Analysis of UAS-LiDAR Ground Points Classification in Agricultural Fields Using Traditional Algorithms and PointCNN. Remote Sensing 15(2): 483.

15. Gharineiat Z, Tarsha Kurdi F, Campbell G (2022) Review of Automatic Processing of Topography and Surface Feature Identification LiDAR Data Using Machine Learning Techniques. Remote Sensing 14(9): 4685.

16. Tarsha Kurdi F, Gharineiat Z, Campbell G, Awrangjeb M, Dey EK (2022) Automatic Filtering of Lidar Building Point Cloud in Case of Trees Associated to Building Roof. Remote Sensing 14(2): 430.

17. Blomley R, Hovi A, Weinmann M, Hinz S, Korpela I, et al. (2017) Tree Species Classification Using within Crown Localization of Waveform LiDAR Attributes. ISPRS Journal of Photogrammetry and Remote Sensing 133: 142-156.

18. Budei BC, St-Onge B, Hopkinson C, Audet FA (2018) Identifying the Genus or Species of Individual Trees Using a Three-Wavelength Airborne Lidar System. Remote Sensing of Environment 204: 632-647.

19. Yang G, Zhao Y, Li B, Ma Y, Li R, et al. (2019) Tree Species Classification by Employing Multiple Features Acquired from Integrated Sensors. Journal of Sensors 2019(3247946): 1-12.

20. Yu X, Hyyppä J, Litkey P, Kaartinen H, Vastaranta M, et al. (2017) Single-Sensor Solution to Tree Species Classification Using Multispectral Airborne Laser Scanning. Remote Sensing 9(2): 108.

21. Ning X, Ma Y, Hou Y, Lv Z, Jin H, et al. (2022) Semantic Segmentation Guided Coarse-to-Fine Detection of Individual Trees from MLS Point Clouds Based on Treetop Points Extraction and Radius Expansion. Remote Sensing 14(19): 4926.

22. Luo H, Khoshelham K, Chen C, He H (2021) Individual Tree Extraction from Urban Mobile Laser Scanning Point Clouds Using Deep Pointwise Direction Embedding. ISPRS Journal of Photogrammetry and Remote Sensing 175: 326-339.

23. Hui Z, Cheng P, Yang B, Zhou G (2022) Multi-Level Self-Adaptive Individual Tree Detection for Coniferous Forest Using Airborne LiDAR. International Journal of Applied Earth Observation and Geoinformation 114: 103028.

24. Zhou T, dos Santos RC, Liu J, Lin YC, Fei WC, et al. (2022) Comparative Evaluation of a Newly Developed Trunk-Based Tree Detection/ Localization Strategy on Leaf-Off LiDAR Point Clouds with Varying Characteristics. Remote Sensing 14(15): 3738.

25. Hell M, Brandmeier M, Briechle S, Krzystek P (2022) Classification of Tree Species and Standing Dead Trees with Lidar Point Clouds Using Two Deep Neural Networks: PointCNN and 3DmFV-Net. PFG 90: 103-121.

26. Mizoguchi T, Ishii A, Nakamura H, Inoue T, Takamatsu H (2017) Lidar-Based Individual Tree Species Classification Using Convolutional Neural Network. In: Remondino F, Shortis MR (Eds.), Munich, Germany, p. 1033200.

27. Marrs J, Ni-Meister W (2019) Machine Learning Techniques for Tree Species Classification Using Co-Registered LiDAR and Hyperspectral Data. Remote Sensing 11: 819.

28. Zou X, Cheng M, Wang C, Xia Y, Li J (2017) Tree Classification in Complex Forest Point Clouds Based on Deep Learning. IEEE Geosci. Remote Sensing Lett 14(12): 2360-2364.

29. Nguyen HM, Demir B, Dalponte M (2019) Weighted Support Vector Machines for Tree Species Classification Using Lidar Data. In Proceedings of the IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium; IEEE: Yokohama, Japan, July 2019; pp. 6740-6743.

30. Tarsha Kurdi F, Amakhchan W, Gharineiat Z (2021) Random Forest Machine Learning Technique for Automatic Vegetation Detection and Modelling in LiDAR Data. IJESNR 28(2): 556234.

31. Schmohl S, Narváez Vallejo A, Soergel U (2022) Individual Tree Detection in Urban ALS Point Clouds with 3D Convolutional Networks. Remote Sensing 14(6): 1317.

32. Windrim L, Bryson M (2020) Detection, Segmentation, and Model Fitting of Individual Tree Stems from Airborne Laser Scanning of Forests Using Deep Learning. Remote Sensing 12(9): 1469.

33. Corte APD, Souza DV, Rex FE, Sanquetta CR, Mohan M, et al. (2020) Forest Inventory with High-Density UAV-Lidar: Machine Learning Approaches for Predicting Individual Tree Attributes. Computers and Electronics in Agriculture 179: 105815.

34. Chen X, Jiang K, Zhu Y, Wang X, Yun T (2021) Individual Tree Crown Segmentation Directly from UAV-Borne LiDAR Data Using the PointNet of Deep Learning. Forests 12(2): 131.

35. Luo Z, Zhang Z, Li W, Chen Y, Wang C, et al. (2022) Detection of Individual Trees in UAV LiDAR Point Clouds Using a Deep Learning Framework Based on Multichannel Representation. IEEE Trans Geosci Remote Sensing 60: 1-15.

36. Olson RS, Moore JH (2019) TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. In: Hutter F, Kotthoff L, Vanschoren J (Eds.), Automated Machine Learning. The Springer Series on Challenges in Machine Learning; Springer International Publishing: Cham, ISBN 978-3-030-05317-8, pp. 151-160.

37. Zöller MA, Huber MF (2021) Benchmark and Survey of Automated Machine Learning Frameworks. Journal of Artificial Intelligence Research 70(2021): 409-472.

38. Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, et al. Efficient and Robust Automated Machine Learning. Part of Advances in Neural Information Processing Systems 28 (NIPS 2015) 9.

39. Olson RS, Bartley N, Urbanowicz RJ, Moore JH (2016) Evaluation of a Tree-Based Pipeline Optimization Tool for Automating Data Science. In Proceedings of the Proceedings of the Genetic and Evolutionary Computation Conference 2016; ACM: Denver Colorado USA, pp. 485-492.

40. Milutinovic M, Baydin AG, Zinkov R, Harvey W, Song D, et al. (2017) End-to-End Training of Differentiable Pipelines Across Machine Learning Frameworks. Neural Information Processing Systems (NIPS) 2017 Autodiff Workshop: The Future of Gradient–based Machine Learning Software and Techniques, Long Beach, CA, US 2017, 5.

41. Nikitin NO, Vychuzhanin P, Sarafanov M, Polonskaia IS, Revin I, et al. (2022) Automated Evolutionary Approach for the Design of Composite Machine Learning Pipelines. Future Generation Computer Systems 127: 109-125.

42. Xin D, Miao H, Parameswaran A, Polyzotis N (2021) Production Machine Learning Pipelines: Empirical Analysis and Optimization Opportunities. In Proceedings of the Proceedings of the 2021 International Conference on Management of Data; ACM: Virtual Event China, pp. 2639-2652.

43. Tao P, Tan K, Ke T, Liu S, Zhang W, et al. (2022) Recognition of Ecological Vegetation Fairy Circles in Intertidal Salt Marshes from UAV LiDAR Point Clouds. International Journal of Applied Earth Observation and Geoinformation 114: 103029.

44. González-Olabarria JR, Rodríguez F, Fernández-Landa A, Mola-Yudego B (2012) Mapping Fire Risk in the Model Forest of Urbión (Spain) Based on Airborne LiDAR Measurements. Forest Ecology and Management 282: 149-156.

45. Elvis - Elevation and Depth - Foundation Spatial Data.

46. (2021) OpenTopography Gisborne, New Zealand 2018-2020.

47. West KF, Webb BN, Lersch JR, Pothier S, Triscari JM, et al. (2004) Context-Driven Automated Target Detection in 3D Data. In: Sadjadi FA (Ed.), Orlando, FL, pp. 133-143.

48. Duran Z, Ozcan K, Atik ME (2021) Classification of Photogrammetric and Airborne LiDAR Point Clouds Using Machine Learning Algorithms. Drones 5(4): 104.

49. Dorofki M, Elshafie A, Jaafar OA, Karim O, Mastura S (2012) Comparison of Artificial Neural Network Transfer Functions Abilities to Simulate Extreme Runoff Data. Int Proc Chem Biol Environ Eng 33.

50. Amakhchan W, Tarsha-Kurdi F, Gharineiat Z, Boulaassal H, el Kharki O (2022) Automatic Filtering of LiDAR Building Point Cloud Using Multilayer Perceptron Neuron Network.; Istanbul, Turkey (HYBRID).

51. Sklearn.Neural_network.MLPClassifier.

52. González-Camacho JM, Ornella L, Pérez-Rodríguez P, Gianola D, Dreisigacker S, et al. (2018) Applications of Machine Learning Methods to Genomic Selection in Breeding Wheat for Rust Resistance. The Plant Genome 11(2): 170104.

53. González-Camacho JM, Crossa J, Pérez-Rodríguez P, Ornella L, Gianola D (2016) Genome-Enabled Prediction Using Probabilistic Neural Network Classifiers. BMC Genomics 17: 208.

54. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, et al. Scikit-Learn: Machine Learning in Python. MACHINE LEARNING IN PYTHON 6.

55. Thomas H, Goulette F, Deschaud JE, Marcotegui B, LeGall Y (2018) Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods. In Proceedings of the 2018 International Conference on 3D Vision (3DV); IEEE: Verona, pp. 390-398.

56. Weinmann M, Jutzi B, Hinz S, Mallet C (2015) Semantic Point Cloud Interpretation Based on Optimal Neighborhoods, Relevant Features and Efficient Classifiers. ISPRS Journal of Photogrammetry and Remote Sensing 105: 286-304.

57. Dey EK, Tarsha Kurdi F, Awrangjeb M, Stantic B (2021) Effective Selection of Variable Point Neighbourhood for Feature Point Extraction from Aerial Building Point Cloud Data. Remote Sensing 13(8): 1520.

58. Tarsha Kurdi F, Awrangjeb M (2020) Comparison of LiDAR Building Point Cloud with Reference Model for Deep Comprehension of Cloud Structure. Canadian Journal of Remote Sensing 46(5): 603-621.

59. Vega C, Hamrouni A, El Mokhtari S, Morel J, Bock J, et al. (2014) PTrees: A Point-Based Approach to Forest Tree Extraction from Lidar Data. International Journal of Applied Earth Observation and Geoinformation 33: 98-108.

60. Kwak DA, Lee WK, Lee JH, Biging GS, Gong P (2007) Detection of Individual Trees and Estimation of Tree Height Using LiDAR Data. Journal of Forest Research 12: 425-434.

61. Mäyrä J, Sarri SK, Kivinen S, Tanhunpaa T, Hurskainen P, et al. (2021) Tree Species Classification from Airborne Hyperspectral and LiDAR Data Using 3D Convolutional Neural Networks. Remote Sens Environ 256: 112322.