International Journal of Geo-Information

MDPI

Article

Machine Learning-Based Supervised Classification of Point Clouds Using Multiscale Geometric Features

Muhammed Enes Atik *, Zaide Duran and Dursun Zafer Seker

Department of Geomatics Engineering, Istanbul Technical University, Maslak, Istanbul 34469, Turkey; duranza@itu.edu.tr (Z.D.); seker@itu.edu.tr (D.Z.S.)
* Correspondence: atikm@itu.edu.tr

Abstract: 3D scene classification has become an important research field in photogrammetry, remote sensing, computer vision and robotics with the widespread usage of 3D point clouds. Point cloud classification, called semantic labeling, semantic segmentation, or semantic classification of point clouds is a challenging topic. Machine learning, on the other hand, is a powerful mathematical tool used to classify 3D point clouds whose content can be significantly complex. In this study, the classification performance of different machine learning algorithms in multiple scales was evaluated. The feature spaces of the points in the point cloud were created using the geometric features generated based on the eigenvalues of the covariance matrix. Eight supervised classification algorithms were tested in four different areas from three datasets (the Dublin City dataset, Vaihingen dataset and Oakland3D dataset). The algorithms were evaluated in terms of overall accuracy, precision, recall, F1 score and process time. The best overall results were obtained for four test areas with different algorithms. Dublin City Area 1 was obtained with Random Forest as 93.12%, Dublin City Area 2 was obtained with a Multilayer Perceptron algorithm as 92.78%, Vaihingen was obtained as 79.71% with Support Vector Machines and Oakland3D with Linear Discriminant Analysis as 97.30%.

Keywords: machine learning; point cloud; geometric features; multiscale; classification; LiDAR

Citation: Atik, M.E.; Duran, Z.; Seker, D.Z. Machine Learning-Based Supervised Classification of Point Clouds Using Multiscale Geometric Features. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 187. https://doi.org/10.3390/ijgi10030187

Academic Editor: Wolfgang Kainz

Received: 10 December 2020
Accepted: 19 March 2021
Published: 21 March 2021

1. Introduction

Information extraction from 3D data has become an important field of research in photogrammetry, remote sensing, computer vision and robotics, with the increasing utility of 3-dimensional (3D) point clouds [1]. Scanning an object with a laser generates 3D information about the shape of the object. This is the basic principle of creating depth perception for machines or 3D machine vision [2]. Point clouds contain information such as 3D position information, density and color. The grouping of points with similar singular properties under a meaningful cluster is called classification. Problems of point cloud classification are insufficiency of existing databases, high hardware requirements for data processing and insufficient methods in terms of accuracy and speed.

Traditionally, point cloud classification relies on defining a set of discriminatory rules to distinguish points for each class [3]. Although discriminatory rules are effective for controlled environments, methods still have inherent limitations when dealing with complex data involving uncertainty and complex relationships between classes. Therefore, point cloud classification for these complex data cannot be handled by combining simple decision rules. Machine learning is a powerful mathematical tool that can be used to classify complex point clouds. In machine learning algorithms, classification rules are learned automatically using training data instead of pre-determined arbitrary parameters. Machine learning and automatic feature selection avoid most of the tedious design and programming work involved in a traditional classification methodology. Thus, these methods are more convenient than traditional classification methods due to their effectiveness for complex data consisting of multiple object types [4]. To accurately predict the semantic classification

algorithms, the label of each point in the 3-dimensional point cloud and the geometry of the point cloud must be correctly defined [5].

This study aims to examine the effects of geometric properties produced at different scales from point clouds on classification accuracy. A comprehensive examination has been realized to determine the support area size and to select the appropriate geometric features. It is aimed to better understand the classification performance of machine learning methods on point clouds of different densities and scales. The significance of geometric properties depending on the data and scale was examined. The datasets used in the study are the ISPRS 3D Semantic Labeling Contest Dataset of Vaihingen [6], Dublin City [7] and Oakland3D [8] datasets. The support areas of each point of the point clouds were calculated in five different scales as 0.5 m, 1 m, 1.5 m, 2 m and 3 m, respectively. Two regions with different sizes of the Dublin dataset were selected for testing, whereas the entire test part of the Vaihingen dataset and Oakland3D dataset were used. In addition to the overall accuracy of the methods, precision, recall and F1 score values were also calculated for each class. Therefore, the effect of different support radii on the extraction of the classes has been examined in detail and the machine learning methods were compared. Algorithms chosen for comparison and evaluation are Random Forest (RF), Linear Discriminant Analysis (LDA), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Multi-Layer Perceptron (MLP), decision tree (DT), Support Vector Machines (SVM) and k-nearest neighbors (KNN). The effects of geometric features change depending on the support radius. To determine this change, feature importance was calculated with the RF method. The Python programming language and open-source Cloud Compare software were used in the study. Multi-scale point cloud classification has been carried out using popular machine learning algorithms cited in the literature. The utility of geometric features in point cloud supervised classification was investigated. The results obtained are presented in the form of tables and images. The flowchart of this applied approach is shown in Figure 1.

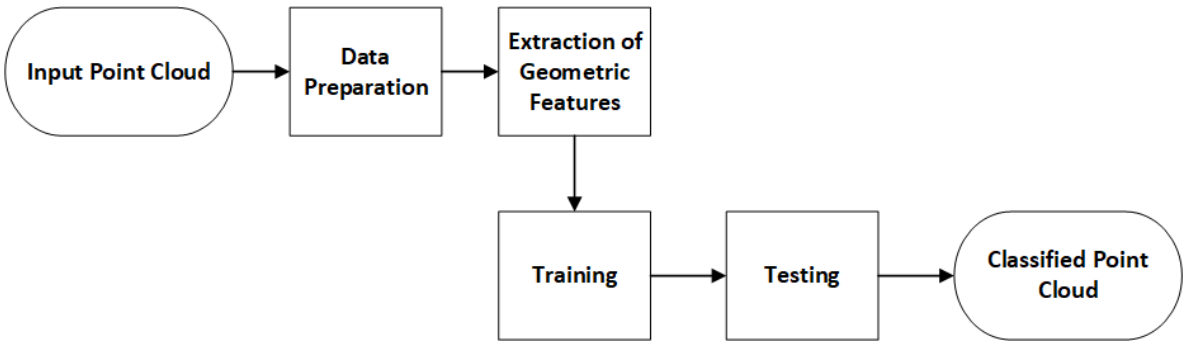

Figure 1. Flowchart of applied approach in the study.

2. Related Works

Supervised learning approaches that assign labels individually to each point in the point cloud can often be chosen. Weinmann et al. [9] examined the use of different features for point cloud classification and the most appropriate support selection and different classifiers. The features that mostly affect the accuracy among produced features were determined by using feature selection methods. The Oakland 3D Point Cloud Dataset and Paris-rue-Lille Dataset, which are mobile laser scanning data, were used as the dataset. Random Forest was found to be the most effective classifier as a result of the evaluation. Vosselman et al. [10] used 19 geometric features that were extracted from the point cloud in the study. In the proposed method, the point cloud was divided into certain groups and classified. The Vaihingen and Rotterdam datasets of International Society for Photogrammetry and Remote Sensing were used as the dataset. Conditional Random Field algorithm was selected as the classification method. Resultantly, group-based classification exerted better results than point-based classification. The landslide was investigated by using point

clouds in [11]. Objects were specified in the point cloud for object-based classification. Features were extracted from the point cloud to identify these objects. Classification was conducted via Support Vector Machines (SVM) and Random Forest (RO) methods using the extracted features. Seven classes were classified: landslide slope, eroded area, sediment, medium and high vegetation, small grasses, high grasses and rocks. In the study conducted by Belgui et al. [12], the buildings were detected from the point cloud which was divided into three sub-classes; small buildings, apartment buildings and industrial buildings. In the study, the RF algorithm was trained using the features extracted from the point cloud. The extracted features were more related to height. Guo et al. [4] proposed an ensemble-learning algorithm named JointBoost for point cloud classification. Ensemble methods aim to create a strong classifier by combining more than one weak classifier. In this study, five classes were classified as building, land, vegetation, power line and power line poles. 26 features were extracted from the point cloud for each point. The proposed approach had two stages. In the first stage, classification was realized with the JointBoost method by using the 17 most effective features. In the next stage, unreliable or misclassified points were again classified by the k-NN method. Landslide regions were determined in the point cloud in [13]. For this purpose, Digital Terrain Model (DTM) produced the point cloud and aerial photographs were used. Firstly, 52 pixel-based features such as slope gradient, plan curvature, roughness and sky view factor were determined on DTM. These features were then transferred to the objects determined by Object-Based Image Analysis (OBIA). An RF-based feature selection algorithm was performed prior to classification because of the large number of features. RF and SVM algorithms were used as classifiers. The study proposed by Plaza-Leiva et al. [14] focused on the efficiency of the spatial shape features obtained from the covariance analysis for improving computational load and accuracy of the supervised learning classification. For this purpose, eigenvalues were calculated for each point from the point cloud. The classification was based on three classes: city buildings, natural vegetation and artificial vegetation. Four different supervised learning algorithms were compared: SVM, artificial neural network, Gaussian operation and Gaussian mixing models. In [15], a supervised classification was performed in both residential areas and forest areas. Five geometric properties were used from the point cloud: linearity, planarity, sphericity, horizontality and height change. SV machines, RF, logistic regression and linear discriminant analysis were chosen as classification algorithms. As a result of classifications with different support radius, an accuracy of approximately 80% for the residential area and 93% for the forest area has been obtained by using RF. In another study [16], supervised classification of point clouds was done using geometric properties. In the study, four different datasets, including color information of points besides 15 geometric features, were used. Ground, high vegetation, building, road, car and human-made objects classes were extracted with RF and Boosted Tress methods. A photogrammetric point cloud was used for training and testing. In the Lin et al. [3] study, the aerial point cloud was classified according to three geometric properties, namely linearity, planarity and sphericity, by using SVM. In the method proposed, it was aimed to increase the classification accuracy by weighting the covariance matrix. The targeted classes were divided into buildings and non-building structures.

As a powerful controlled statistical method, machine learning is the method that can be used to classify point clouds. When a point is defined by distinctive geometric properties, machine learning is used to predict the class of a point. Since the distinctiveness of geometric features is strongly dependent on the support areas of these 3D points considered for feature extraction, support area size and feature selection are important problems. With this study, it is evaluated that it can contribute to the gap about scale and feature selection in the literature. The effects of geometric properties obtained from different scales on current datasets were investigated. Some criteria based on results have been proposed for the choice of geometric feature and optimum support size.

3. Data and Methodology

3.1. Data Used

The proposed approach was evaluated for three different point cloud datasets of Dublin City, Vaihingen and Oakland3D that included urban areas. The Dublin data set was produced in 2015 by the Urban Modeling Group at University College Dublin (UCD) by scanning with ALS in Dublin City Center. 260 million out of 1.4 billion points have been labeled. The density of the point cloud varied from 250 to 348 points/m^2. The point cloud consisted of 13 regions in total. In this study, since it was not possible to use the whole dataset due to hardware deficiencies, only two regions were selected and used. These two study regions covered approximately 20 million points. A training area (Figure 2) that covered approximately 12 million points was selected. In this area, points labeled as "undefined" were eliminated and for the rest of this process, 970,000 points were selected randomly from the dataset used for training to provide an advantage in efficiency and speed. Two test sites of Dublin City Area 1 and Dublin City Area 2 (Figure 2) with approximately 1.6 million points and 7 million points were defined for testing. While selecting Dublin City Area 2, attention was paid to cover Dublin City Area 1 as well.

Figure 2. Training and test samples from the Dublin City dataset.

The Vaihingen dataset (Figure 3) consisted of training that had 753,859 points and testing parts that had 411,722 points. The Vaihingen dataset included 9 classes, namely powerline, low vegetation, impervious surface, car, fence/hedge, roof, façade, shrub and tree. The classes were reduced into three classes to simulate this dataset to the Dublin dataset for an accurate comparison. Power line, car and fence/hedge classes were eliminated as they had few points. The ground surface class was created by combining low vegetation and impervious surface. Building class was formed by combining roof and façade. The vegetation class was created by combining tree and shrub.

The Oakland dataset, one of the most used MLS datasets, was also selected for the application. This dataset was obtained with a mobile platform and covers the urban environment. The Oakland dataset consists of 36,932 training points, 91,579 validation points and 1.3 million testing points, which include 5 classes, namely ground, vegetation, façade, wire and pole/trunk. The wire and pole/trunk classes were removed, so they contain a few points. The façade class was renamed as building. Training (Figure 4) and testing (Figure 5) parts were used in the study. All datasets had ground truth. The distribution of the training and test data of the datasets are given in Table 1.

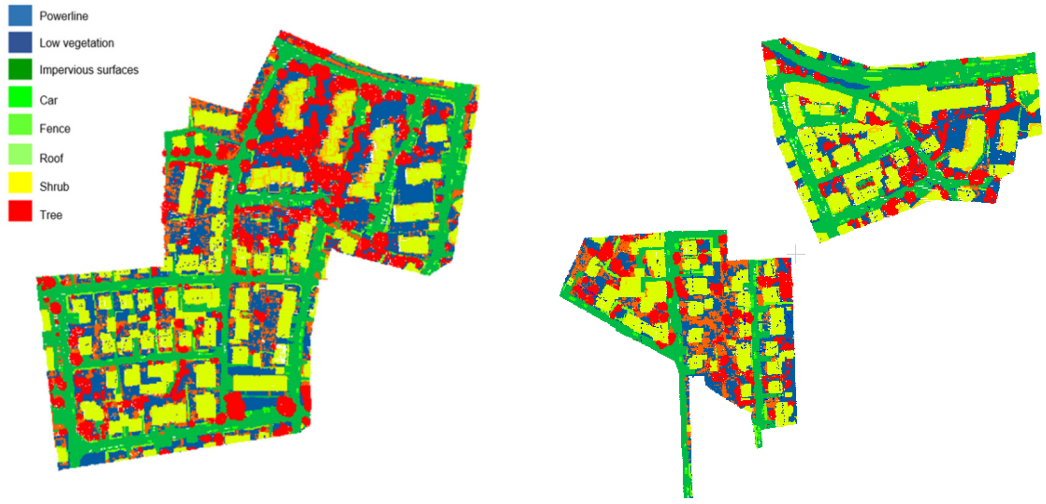

Figure 3. Training and test samples from Vaihingen dataset. (**a**) Training sample. (**b**) Test sample.

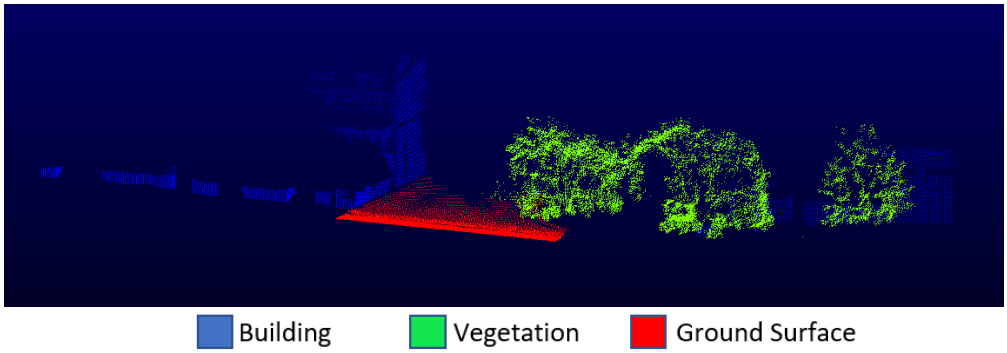

Figure 4. Training samples from the Oakland dataset.

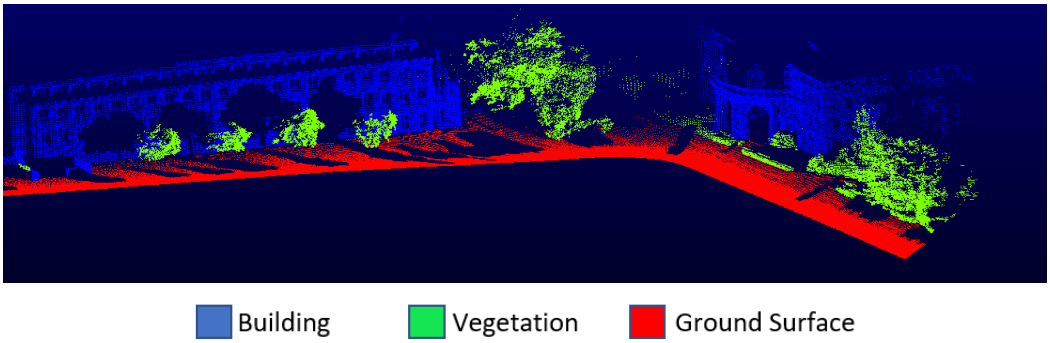

Figure 5. Testing samples from the Oakland dataset.

Table 1. Class distribution of the used datasets.

Dataset	Class	Training	Test
Dublin City Area 1	Building	400,000	485,355
	Vegetation	70,000	360,618
	Ground	500,000	734,055
	Total	970,000	1,580,028
Dublin City Area 2	Building	400,000	2,876,063
	Vegetation	70,000	881,150
	Ground	500,000	3,245,362
	Total	970,000	7,002,575
Vaihingen	Building	179,295	120,272
	Vegetation	182,778	79,044
	Ground	374,573	200,676
	Total	736,646	399,992
Oakland 3D	Building (Façade)	4713	111,112
	Vegetation	14,441	267,328
	Ground	14,121	934,146
	Total	33,275	1,312,583

The Dublin City and Vaihingen datasets had relabeled land-use types consisting of 3 classes; building, vegetation and ground surface. Firstly, support (S) areas of 0.5 m, 1 m, 1.5 m, 2 m and 3 m were created for each point. 13 geometric features were calculated in each support created. Geometric features were expressed as functions of the eigenvalues of the covariance matrix of the set of points located in a certain support area. Feature space was prepared for each point with the calculated geometric features and z coordinate values of the points. Thus, it was used as input data for machine learning algorithms.

3.2. Random Forest (RF)

RF [17] is an improved version of bagging that creates a large collection of uncorrelated trees and then averages them. It is quite similar to boosting the performance of random forests in many problems and is easier to train and adjust [18]. Each tree in the random forest gives a class estimate, and the top voted class becomes the model's prediction. In the bagging algorithm, multiple bootstrap training data sets are created from the original training data set to train a classifier, and a training data set is assigned to each tree. There are two reasons to use bagging. First, the use of bagging appears to increase accuracy while using random features. Second, bagging can be used to give estimates of power and correlation as well as estimates of the generalized error (PE^*) of the combined tree community [17].

Two parameters are required to generate a tree with the RF classifier. These parameters are the number of variables used in each node and the number of trees to develop to determine the best split. Boot samples are created from 2/3 of the training data set. The remaining 1/3 of the training data set, also called out-of-bag (OOB) data, is issued to test errors. The error obtained from this process is called the generalized error. Generalized error calculation is shown in Equation (1):

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \quad (1)$$

where $mg()$ refers to the margin function. Margin measures how much the average number of votes in (X, Y) for the correct class exceeds the average rating for any other class. The more reliably the classification can be performed, the larger the margin [17].

3.3. Naïve Bayes (NB)

Classifiers work by computing one separator function for each class and assigning a sample to the class in which the function takes its largest value [19]. For example, assume

that a is a vector of attributes, as in typical classification practices. In the example, let v_{jk} be the value of the attribute A_j, $P(X)$ represents the probability of X and $P(Y \mid X)$ the conditional probability of X given Y. Then, a possible set of separator functions can be expressed as Equation (2):

$$f_i(E) = P(Ci) \prod_{j=1}^{a} P(A_j = v_{jk} | C_i) \tag{2}$$

The classifier obtained using this set of discriminant functions and predicting the related probabilities in the training set is usually called the Naïve Bayes classifier [20]. Naïve Bayes 'classifier is a probabilistic classifier based on Bayes' theorem. Its starting point is Bayes' conditional probability theorem and it examines the probabilistic relationship between a particular data point x and class C [21].

$$P(C \,|\mathrm{x}) = \frac{P(x \,|\mathrm{C})}{P(x)} \tag{3}$$

Although it has different types, it is the most widely used Gauss Naïve Bayes classifier. Gauss Naïve Bayes (GNB) applies classification by assuming the probability of the properties to be Gauss [22]. The formulation is as Equation (4):

$$P(x_i \,|\mathrm{C}) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(x_i - \mu_C)^2}{2\pi\sigma_C^2} \right) \tag{4}$$

3.4. Multilayer Perceptron (MLP)

MLP is a sensor with a single weight layer that can only operate on linear functions of the input. It cannot give successful results on nonlinear functions. Multi-layer perceptron can solve such nonlinear problems if used for classification [23]. Multi-layer Sensor is a supervised learning algorithm that learns a function by training on a dataset. MLP can learn a nonlinear function approach for classification or regression. There may be one or more nonlinear layers, called hidden layers, between the input and output layer. Therefore, it is different from logistic regression [24]. In MLP, there are transitions between layers called forward and backward propagation. In the forward propagation phase, the output of the network and the error value are calculated. In the backpropagation phase, the connection weight values between the layers are updated to minimize the calculated error value.

The input layer is fed with the input x value. The "activation" propagates in the forward direction and z_h values are calculated in the hidden layers. Each hidden layer unit is a detector on its own and applies the nonlinear sigmoid function to its weighted sum:

$$z_h = sigmoid\left(w_h^T x\right) = \frac{1}{1 + exp\left[-\left(\sum_j^d w_{hj} x_j + w_{h0}\right)\right]} \tag{5}$$

To calculate y_i values in the output layer, the sensors in this section use the values calculated in hidden layers as input values [23].

$$y_i = v_i^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0} \tag{6}$$

3.5. Logistic Regression

In logistic regression, the probability that the output variable belongs to the appropriate class is calculated [25]. Logistic regression [26] establishes a linear transformation between the output variable and input variables. However, a linear transformation is performed between input variables and probabilities of the output categorical variable, instead

of between the output and the input variables. Input variables need not be continuous, normally distributed or independent. The result is the probability of an observation for a category, not a class. The mathematical model for the logistic regression classifier can be expressed as:

$$\ln \frac{P(Y=i)}{P(Y=C)} = \alpha_i + \sum_{j=1}^{p} \beta_j x_j \tag{7}$$

where Y is the output variable, α_i and β_j are the coefficients of the model, and $x_1, x_2, \ldots, x_p$ are the covariates. The basic mathematical concept that defines logistic regression is logit, which can be defined as the natural logarithm of a probability ratio. Each observation is assigned to the class of maximum probability:

$$P(Y=i|X) = \frac{e^{\alpha_i + \sum_{i=1}^{j=p} \beta_j X_j}}{1 + e^{\alpha_i + \sum_{i=1}^{j=p} \beta_j X_j}} \tag{8}$$

In general, logistic regression is well suited for explaining and testing hypotheses about relationships between a categorical output variable and one or more categorical variables [27].

3.6. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) [28] is the oldest classifier currently in use, is a linear transformation that calculates the directions of the axis that best distinguishes multiple classes. If we define this direction as w, the data is projected onto the defined w direction. Thus, the sample data of the two classes are tried to be separated as much as possible.

$$z = w^t x \tag{9}$$

z is the projection of data (x) onto w. After the projection, to separate the classes well, it is aimed to keep the average of the samples belonging to the classes as far from each other as possible and to distribute the class samples to as small an area as possible. In LDA, it is aimed to have a maximum ratio of within-class (S_i) and between-class (S_B) scatter matrices ($R = S_i/S_B$).

$$S_i = \sum_t r_i^t \left(x^t - m_i\right)\left(x^t - m_i\right)^T \tag{10}$$

$$S_B = \sum_{i=1}^{K} N_i (m_i - m)(m_i - m)^T \tag{11}$$

The algorithm tries to assign elements to each class, maximizing R [23]. The solution is the largest eigenvectors of $S_W^{-1} S_B$, where S_w is the sum of the within-class scatter matrix. LDA assumes that attributes (input or explanatory variables) are continuous and normally distributed, while the dependent variable, which is the output value (such as a class), is categorical [15].

3.7. Decision Tree (DT)

Decision tree (DT) is one of the algorithms that divides the input space into parts and calculates parameters for each part. Decision tree is a non-parametric classification and regression algorithm. Each node in the decision tree is associated with a part, and nodes divide each part into sub-parts. If the size of the decision tree is not predetermined, it can be considered as an algorithm that is non-parametric [29].

Besides being easy to apply, decision tree has some disadvantages. It can create a complex tree on the data. Decision trees can be unstable because small changes in data can result in producing a completely different tree. However, it is still a useful algorithm due to computational limitations.

3.8. Support Vector Machines (SVM)

Support Vector Machines (SVM) [30] is a supervised machine learning algorithm used for both classification and regression. The aim of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that has the maximum distance between data points of both classes and classifies the data points separately. The optimal hyperplane can be obtained by using Equation (12). For a given set of a sample $x_i (i = 1, 2, \ldots, N)$:

$$f(x) = w^T x + b = \sum_{j=1}^{N} w_j x_j + b = 0 \quad (12)$$

where w is an N-dimensional vector and b is a scalar, and they are used to define the hyperplane. There are two hyperplanes that separate the samples and are not points between them, to separate samples in a dataset linearly [31]. In this study, a linear SVM algorithm optimized with Stochastic Gradient Descent (SGD) is used. SGD is one of the most popular optimization algorithms used in machine learning in large data sets due to the ability to efficiently optimize the entire training set depending on the number of data epochs [32]. It aims to approximate the gradient of the objective function using small randomly selected subsets of training examples. SGD chooses a point in each iteration or a set of points based on batch size to reduce a large processing load, rather than using all the data for learning. Depending on the amount of data in SGD, a balance is struck between the accuracy of the weight update and the time it takes to perform an update. In one iteration, the weights are updated according to each random point or set of points selected. Thus, the gradient of the sample i selected in t iteration $\nabla E\ (W_t, x_i, y_i)$ is calculated instead of the true gradient (∇E) [33].

3.9. K-Nearest Neighbor (KNN)

K-nearest neighbor (KNN)-based classification is a type of sample-based learning or non-generalized learning. It does not attempt to create a general internal model [34]. The nonparametric k-nearest neighbor algorithm is not constrained by a fixed number of parameters. Since KNN does not contain parameters, it generates and applies a function dependent on training data. For each data in an input data set X, k's closest neighbors are found. Neighboring points are weighted in inverse proportion to their distance from the query point. Afterward, the most common class in the neighborhood is assigned as the class x data [23].

3.10. Extraction of Geometric Features

Geometric properties are useful for explaining the local geometry of points. These geometric features are nowadays widely applied in LiDAR data processing. It is aimed to improve the accuracy values by extracting these geometric features in multiple scales rather than on a single scale. Geometric features are calculated by the eigenvalues (λ_1, λ_2, λ_3) of the eigenvectors (v_1, v_2, v_3) derived from the covariance matrix of any point p of the point cloud:

$$cov(S) = \frac{1}{S} \sum_{p \in S} (p - \overline{p}\,)(p - \overline{p})^T \quad (13)$$

where $\overline{p}$ is the centroid of the support S. Many values are calculated using eigenvalues: the sum of eigenvalues, omnivariance, eigenentropy, anisotropy, planarity, linearity, surface variation, sphericity and verticality. Calculated values are normalized. Since the datasets used contained only geometric information (3D coordinates), only geometric features were used in the study (Table 2).

Table 2. Geometric features used in the study.

Feature	Explanation
Sum of eigenvalues	$\lambda_1 + \lambda_2 + \lambda_3$
Omnivariance	$\sqrt[3]{\lambda_1 \lambda_2 \lambda_3}$
Eigenentropy	$\sum_{i=1}^{3} \lambda_i \ln \lambda_i$
Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
Surface variation	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
Sphericity	λ_3/λ_1
Verticality	$1 - \lvert\langle[0\,0\,1], \lambda_3\rangle\rvert$
Height value	Z_i
Roughness Normal change rate Volume density	Other features

The relevant geometric properties of any point in the 3D point cloud were based on the support of the point. Support selection is an important issue as the distinctiveness of the geometric features depends on the relevant support considered for feature extraction [9].

3.11. Experiment

In the first step of the experiment, 13 different feature geometric features were calculated for each data set. Geometric properties were obtained by using spheres with radii of 0.5 m, 1 m, 1.5 m, 2 m and 3 mto apply the classification at different scales.

- $R_{0.5}$; represents the support calculated with a radius of 0.5 m.
- R_1; represents the support calculated with a radius of 1 m.
- $R_{1.5}$; represents the support calculated with a radius of 1.5 m.
- R_2; represents the support calculated with a radius of 2 m.
- R_3; represents the support calculated with a radius of 3 m.

First, the classes of the data were arranged and were reformatted according to algorithms. In the Vaihingen dataset, since there were few points in the power line, car and fence/hedge classes, they were deleted from the dataset. However, wire and pole/trunk classes also had fewer points in the Oakland3D dataset, so they were removed from the dataset.

In the study, point clouds were classified using eight different machine-learning algorithms: LDA, RF, GNB, LR, MLP, DT, SVM and KNN. Scikit-learn, a machine learning library prepared for the Python programming language, was used to implement the methods. The optimized parameters for all methods were determined and were unchanged during the study. In the study, a computer was used with I7 7700HQ 2.8GHZ, 16GB RAM, GTX1050Tİ 4GB graphics card.

Four metrics were used: precision, recall, F1 score for each class and overall accuracy for evaluating of the methods. Precision measures the proportion of points classified as positives (Equation (14)). Recall measures the proportion of positives that are true positives (Equation (15)) [23]. *F*1 score is a function of precision and recall (Equation (16)) [35]. Overall accuracy measures the proportion of points correctly classified. Accuracy metrics are obtained using confusion matrices generated as a result of classification. All accuracy analyzes were carried out in the Python programming language.

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1\ score = 2\frac{Precision\ .\ Recall}{Precision +\ Recall} \tag{16}$$

True positive (*TP*) is the number of points that have the same label in predicted and ground truth. False positive (*FP*) refers to the number of points that are predicted positive but their actual label is negative. False negative (*FN*) refers to the number of points that are predicted negative and their actual label is positive [23].

4. Results

The proposed methodology was applied to the three different areas selected from two different datasets. It was aimed to determine the most suitable support radius for each method. The classification process was repeated in the same way for each case. Points that did not have sufficient neighbor points for geometric properties were eliminated from the point cloud.

According to results in Dublin City Area 1, the highest overall accuracy in algorithms other than GNB, KNN and DT was obtained when using a support radius R_3 (Table 3). The RF method had the highest accuracy of 93.12% (Figure 6). The second-highest accuracy belonged to the MLP method with an overall accuracy of 93.07% with R3. The SVM method has the lowest overall accuracy at 83.27% with $R_{0.5}$ in Dublin City Area 1 (Figure 6). According to the results in different scales, the lowest accuracies were obtained in cases where the support radius of 0.5 m was used. The reason for determining a maximum radius of 3 m is that the calculation load and the time spent increase as the radius increases for the calculation of geometric properties.

Table 3. The overall accuracy of the methods in Dublin City Area 1. The highest-obtained accuracies are shown as bold.

Support	LDA	RF	GNB	LR	MLP	DT	SVM	KNN
$R_{0.5}$	84.65	90.76	**89.70**	83.43	85.85	85.79	83.27	85.58
R_1	86.12	92.33	87.90	88.41	91.45	89.50	89.06	89.42
$R_{1.5}$	86.46	92.03	84.36	89.12	91.71	89.50	90.61	**91.51**
R_2	87.00	92.45	88.87	88.67	91.54	**89.65**	89.26	89.53
R_3	**90.33**	**93.12**	86.97	**90.00**	**93.07**	89.16	**90.62**	87.01

In Dublin City Area 2, 6 out of 8 methods presented the best results when using a support radius of 3 m. The GNB and KNN methods achieved the highest overall accuracy using supports R_{05} and $R_{1.5}$, respectively. In the Dublin dataset, GNB and KNN methods were successful with a smaller support radius. MLP and SVM had better overall accuracy with 92.78% (Figure 7) and 91.59%, respectively. The least accurate method was the LDA method with 84.79% overall accuracy (Figure 8). The overall accuracy of the all methods in Dublin City Area 2 was represented in Table 4.

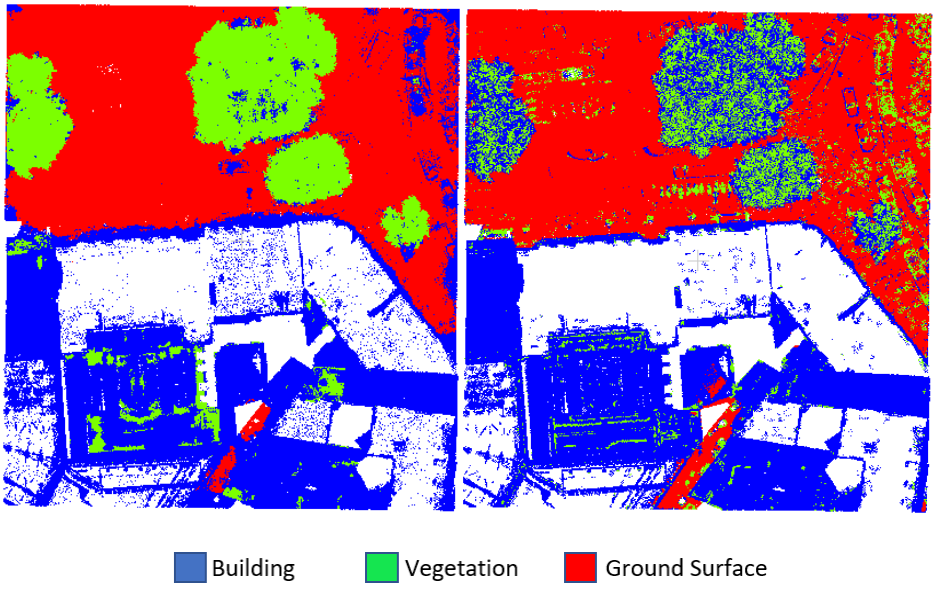

Figure 6. The best result (RF with R_3) and the worst (SVM in $R_{0.5}$) results from Dublin City Area 1. (Result of the RF is on the (**left**); result of the SVM on the (**right**)).

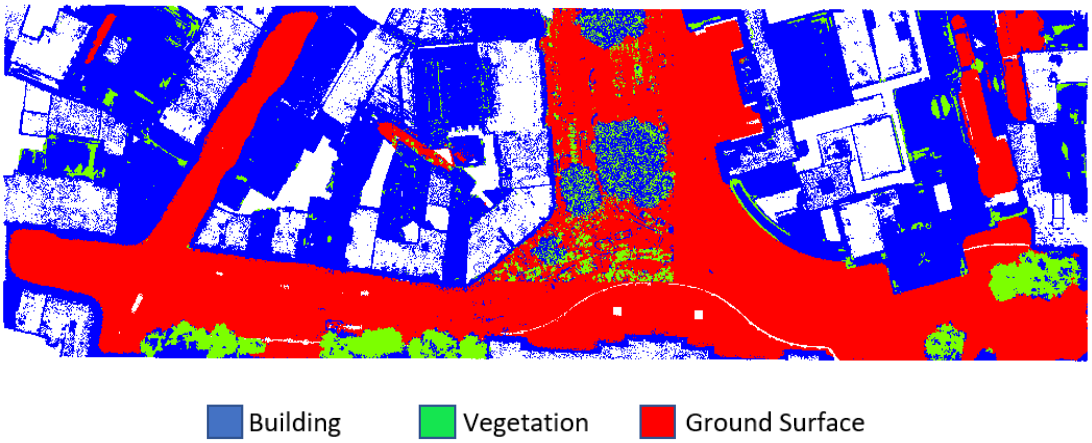

Figure 7. The best classified point cloud from Dublin City Area 2 (MLP with R_3).

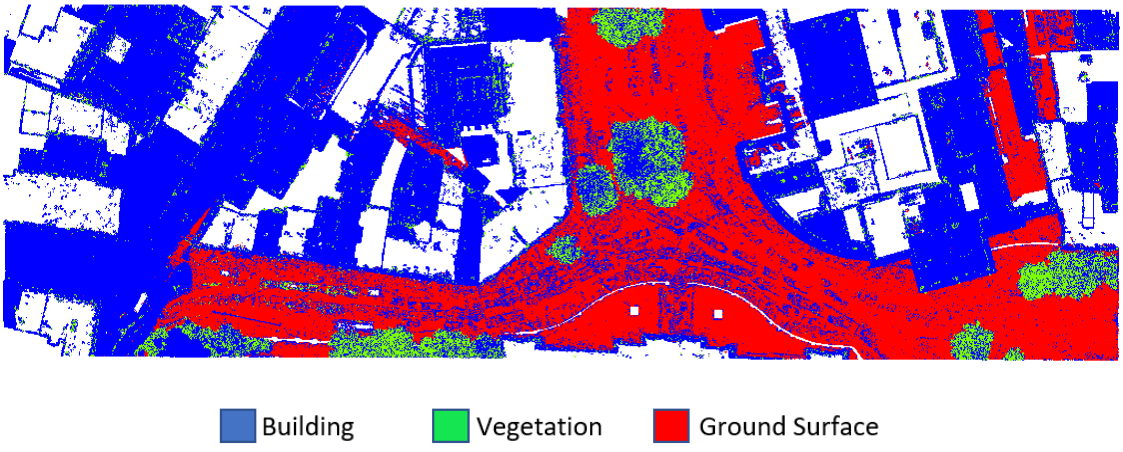

Figure 8. The worst classified point cloud from Dublin City Area 2 (DT with $R_{0.5}$).

Table 4. The overall accuracy of the methods in Dublin City Area 2. The highest-obtained accuracies are shown as bold.

Support	LDA	RF	GNB	LR	MLP	DT	SVM	KNN
$R_{0.5}$	81.61	84.18	**89.79**	88.14	88.70	81.07	88.03	86.91
R_1	81.58	85.15	88.65	90.29	90.65	82.82	88.49	**88.65**
$R_{1.5}$	81.54	85.84	86.88	90.24	92.14	83.51	87.01	88.49
R_2	82.41	86.32	87.07	90.57	91.62	83.14	90.49	85.66
R_3	**84.79**	**88.92**	88.61	**91.10**	**92.78**	**86.54**	**91.59**	83.53

The Vaihingen dataset covered a comparatively smaller area than Dublin City Area 1 and Dublin City Area 2. Geometric features could not be calculated since most of the points in the support of R_{05} did not have sufficient neighboring points. For this reason, $R_{0.5}$ support was not evaluated for the Vaihingen dataset. All methods except KNN had their best results in the R_{15} support. Unlike the Dublin City dataset, the accuracy of all methods decreased with the larger supports (R_2 and R_3) in the Vaihingen data. The two most accurate methods, SVM and MLP, had an overall accuracy of 79.71% and 77.68%, respectively. RF had the lowest accuracy with 68.26% accuracy (Figure 9). The overall accuracy of the all methods in the Vaihingen dataset is represented in Table 5.

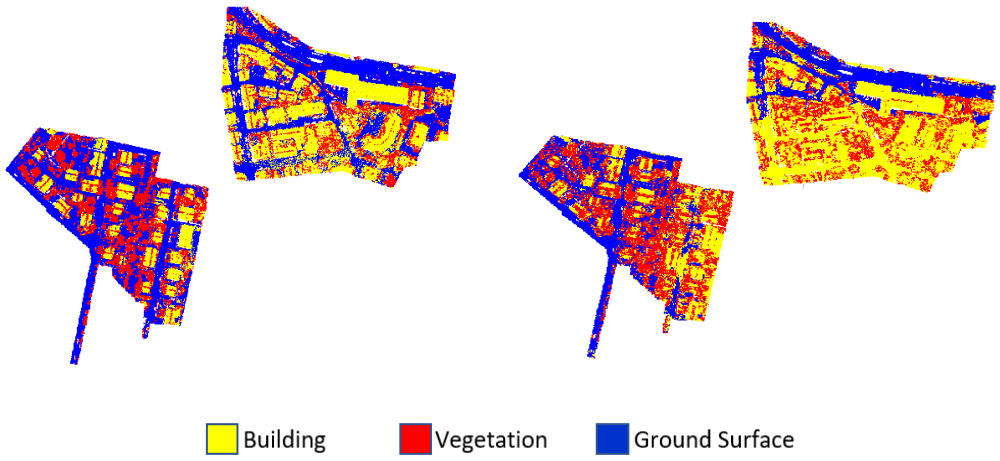

Figure 9. Classified point cloud with the highest accuracy (**left**) and with lowest accuracy (**right**) in the Vaihingen dataset.

Table 5. The overall accuracy of the methods in Vaihingen. The highest-obtained accuracies are shown as bold.

Support	LDA	RF	GNB	LR	MLP	DT	SVM	KNN
$R_{0.5}$	-	-	-	-	-	-	-	-
R_1	69.81	67.07	72.59	69.51	77.21	61.82	77.57	**74.76**
$R_{1.5}$	**70.17**	**68.26**	**75.47**	**70.65**	**77.68**	**62.88**	**79.71**	74.46
R_2	67.39	67.55	74.23	67.53	77.29	62.15	77.78	74.31
R_3	67.24	64.58	71.29	68.66	73.05	60.31	75.35	70.32

The Oakland3D dataset had a similar size to Vaihingen. The highest accuracy values were obtained in R_1 and $R_{1.5}$ supports for all methods except SVM and KNN methods. The highest accuracy was achieved as 97.30% with the LDA method in R_1 support (Figure 10). The LDA and RF methods provided sufficient accuracy for all support sizes. Although the accuracy values decreased slightly as the support size increased, accuracy of around 90% was obtained. The DT method has low accuracy in support sizes except for $R_{1.5}$. 94.12% general accuracy was obtained with the DT method in $R_{1.5}$ support. The

method with the lowest accuracy of 66.78% is the GNB method in R_3 support (Figure 11). General accuracy values of the Oakland version are presented in Table 6.

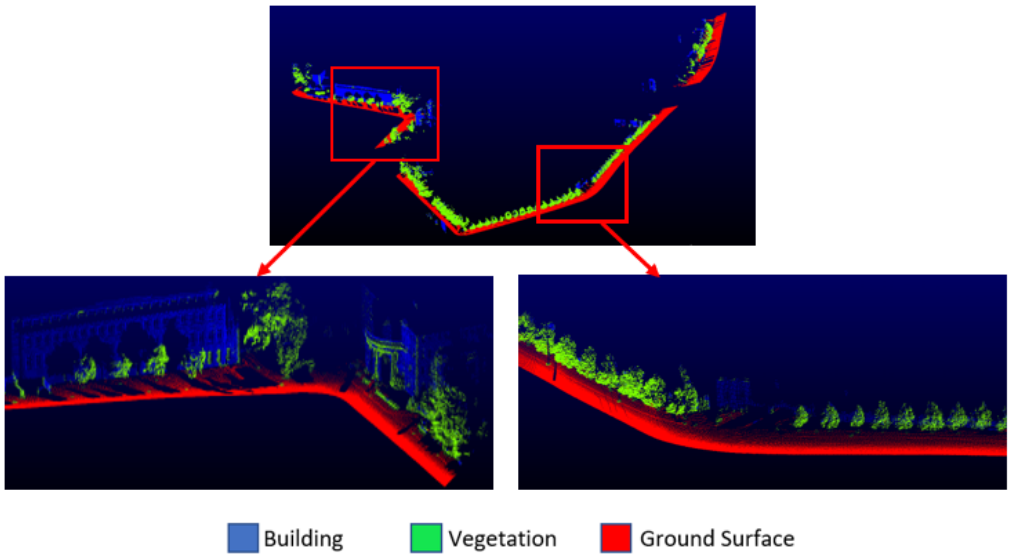

Figure 10. Classified point cloud with the highest accuracy in the Oakland3D (97.30% with LDA in R_1).

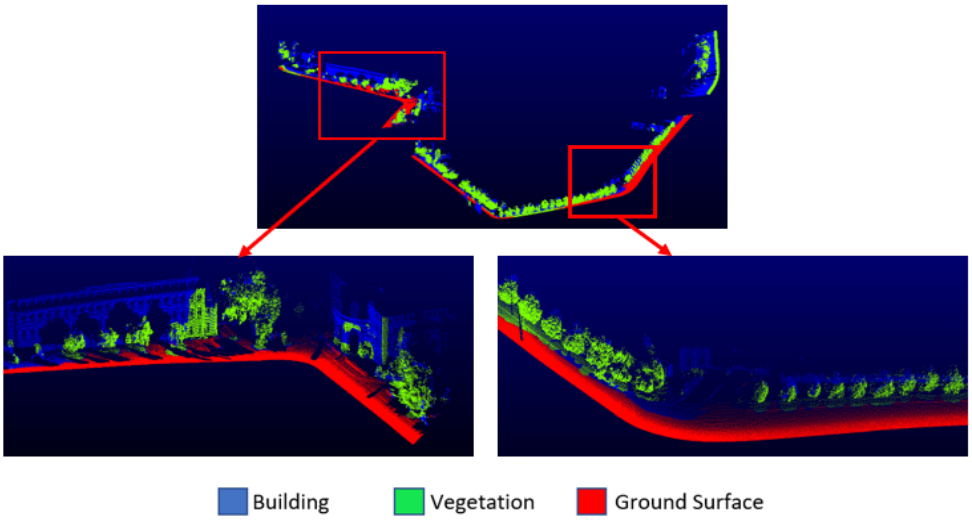

Figure 11. Classified point cloud with the lowest accuracy in the Oakland3D (66.78% with GNB in R_3).

Table 6. The overall accuracy of the methods in Oakland3D. The highest-obtained accuracies are shown as bold.

Support	LDA	RF	GNB	LR	MLP	DT	SVM	KNN
$R_{0.5}$	97.02	95.76	75.40	76.17	81.77	72.85	**73.24**	80.20
R_1	**97.30**	**96.80**	77.59	**78.29**	83.65	75.38	72.72	80.61
$R_{1.5}$	96.34	96.68	**78.01**	77.90	**84.42**	**94.12**	71.78	81.11
R_2	95.24	93.59	72.77	74.49	80.54	76.21	71.13	82.05
R_3	91.94	89.85	66.78	74.07	76.15	87.99	71.44	**83.24**

Precision, recall and F1 score values of each class were also calculated. It was found that in Dublin City Area 1, all classes were detected better as the radius increased. Considering the F1 score, the RF method was the best method for the building class. RF was slightly affected by radius change for the building class. It had an F1 score of 88% for $R_{0.5}$

and 89% for R_3. The KNN method was the most unsuccessful method for the building class. The vegetation class had lower accuracy values in a small support radius. The highest F1 score for vegetation was obtained for RF (91%) in the R_3 support. It has been determined that the KNN method was insufficient for the vegetation class regardless of the support size. KNN generally had a high precision value and low recall value. The opposite was true in the R_3 support. It had a low recall value in the support of $R_{0.5}$ in all methods. In other words, although the methods correctly detected the majority of vegetation class in small supports, most of the points labeled as vegetation belonged to different classes. The ground surface class had been extracted with approximately 100% precision, recall and F1 score in all cases (Figure 12).

Figure 12. Precision, recall and F1 score values of the algorithms for Dublin City Area 1.

In Dublin City Area 2, there were some differences from Dublin City Area 1. While all values increased rapidly in support radius greater than $R_{0.5}$ in Area 1, the values of

metrics at a small radius in Area 2 did not increase in the same way. High metric values were obtained in the support of R_3 for almost all methods. Although precision was not affected by the radius change, recall and F1 score had low values in the support of $R_{0.5}$. MLP and LR (in R_3 support), the best methods for building class extraction, had 97% and 96% F1 scores, respectively. The method with the lowest F1 score was the LDA method with 89% (in R_3 support). Vegetation class was detected with lower success in all supports than other classes. For the vegetation class, RF was the most successful method in terms of F1 score (86% for R_3). KNN was quite insufficient for extraction of vegetation. The highest F1 score was obtained with 72% in $R_{1.5}$ support. Although the ground surface class did not have values close to 100% in Area 2 as in Area 1, because the test data was larger, precision, recall and F1 scores were generally obtained above 90% for all methods (Figure 13).

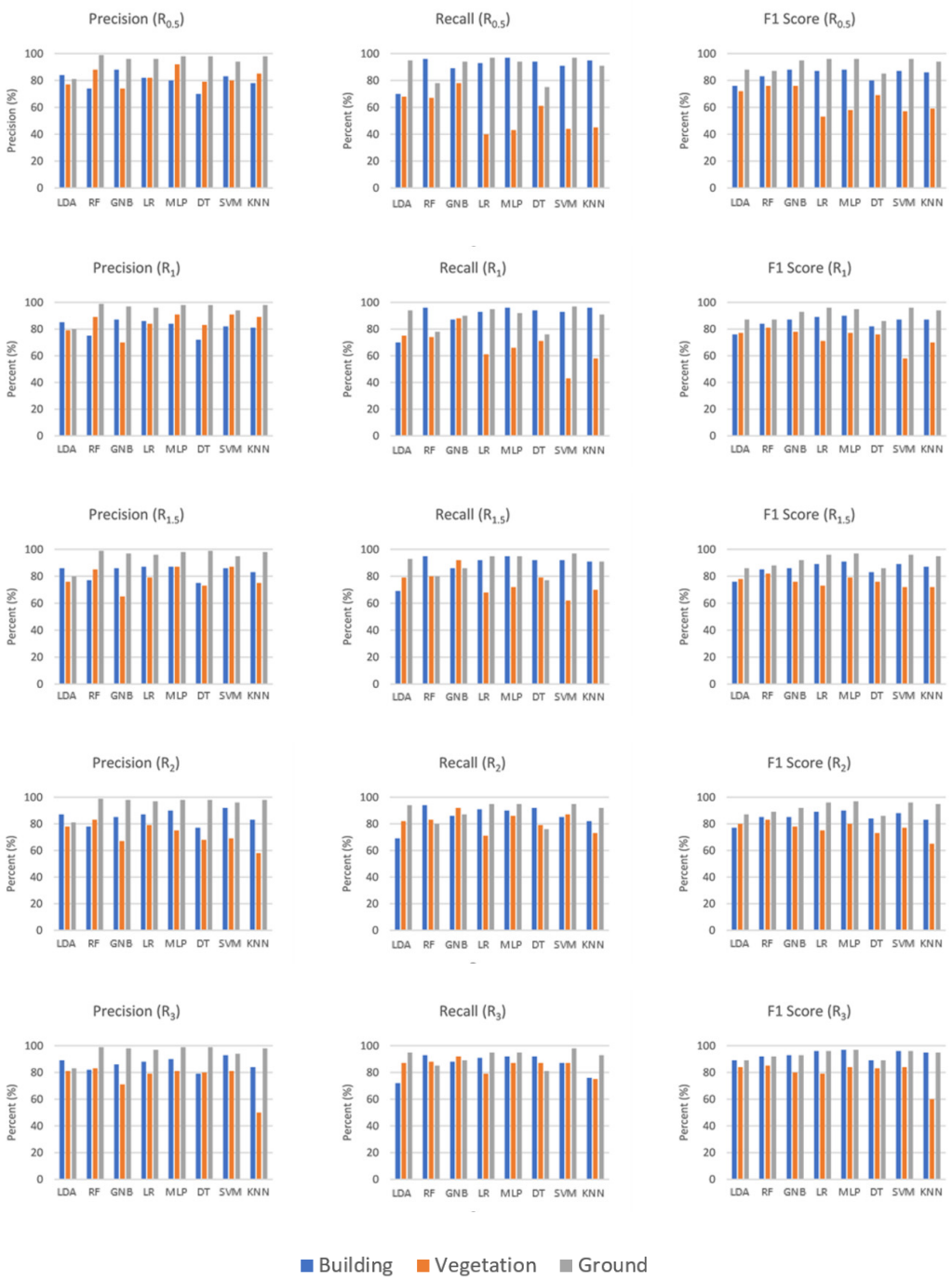

Figure 13. Precision, recall and F1 score values of the algorithms for Dublin City Area 2.

In the Vaihingen dataset, lower metrics were obtained for all methods compared to the Dublin City dataset. In the Vaihingen dataset, recall values were higher than precision values. In other words, although most of the labeled points were labeled correctly, they could not find the points belonging to that class with the same success. For the building class, the SVM method had the highest F1 score value (74% in $R_{1.5}$ support), while the lowest F1 score value was obtained in the DT method (54% in R_3 support). Similar results to the building class were obtained for the vegetation class. MLP (68% in $R_{1.5}$ support) and SVM (68% in $R_{1.5}$ support) have the highest F1 score value. The lowest F1 score (53% in R_3 support) was obtained by the DT method. The methods again extracted the ground surface class better than the other classes. For the ground surface class, SVM (87% F1 score in $R_{1.5}$ support) was the best method while the worst method was DT (68% F1 score in R_3 support). When using large supports (R_2 and R_3) in the Vaihingen dataset, a decrease was observed in all metrics (Figure 14).

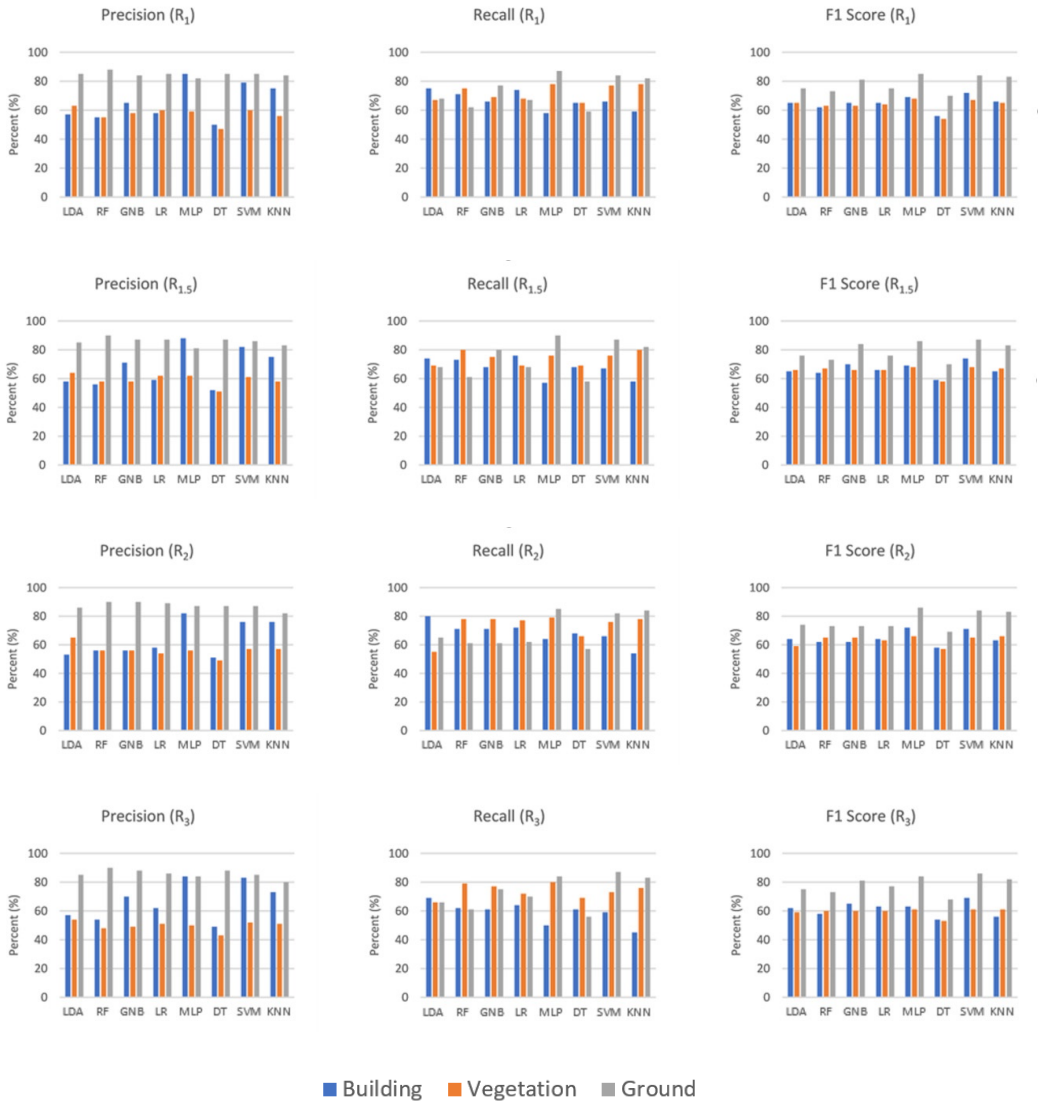

Figure 14. Precision, recall and F1 score values of the algorithms for Vaihingen.

In the Oakland3D dataset, it was determined that better results were obtained in R_1 and $R_{1.5}$ supports. As with the Vaihingen dataset, recall values were higher than precision values at all scales. Especially in R_2 and R_3 supports, a decrease was observed in all accuracy metrics. Similar to other datasets, insufficient accuracy was obtained in R_{05} support. For building class, while the highest F1 score was obtained with LDA (88% in R_1

support), the lowest F1 score was obtained with KNN (36% in $R_{0.5}$ support). For vegetation class, LDA has the highest F1 score (94% in R_1 support), SVM has the lowest F1 score (45% in $R_{0.5}$). The ground surface class was extracted with higher accuracy compared to other classes. When RF has the highest F1 score (%99 in R_1 support), GNB has the lowest F1 score (76% in R_3). The precision, recall and F1 scores of the Oakland3D dataset were shown in Figure 15.

Figure 15. Precision, recall and F1 score values of the algorithms for the Oakland 3D dataset.

Geometric features affected the accuracy at different rates. This situation was defined as feature importance. Some methods calculate the feature importance value. RF is a method to obtain an ordered list of them. The most important feature in both datasets was

the Z coordinates of the points. Other important features for the Dublin City dataset were verticality and volume density. The least important feature was linearity. The roughness feature was also of low importance except for the R_3 support. In the Vaihingen dataset, verticality was the second most important feature. While the least important feature was Linearity, the importance of roughness increased as the support radius grew. Since most of the values were obtained as NaN, no evaluation was made for $R_{0.5}$ in the Vaihingen dataset. Feature importance values calculated for each study area were presented in Figure 16.

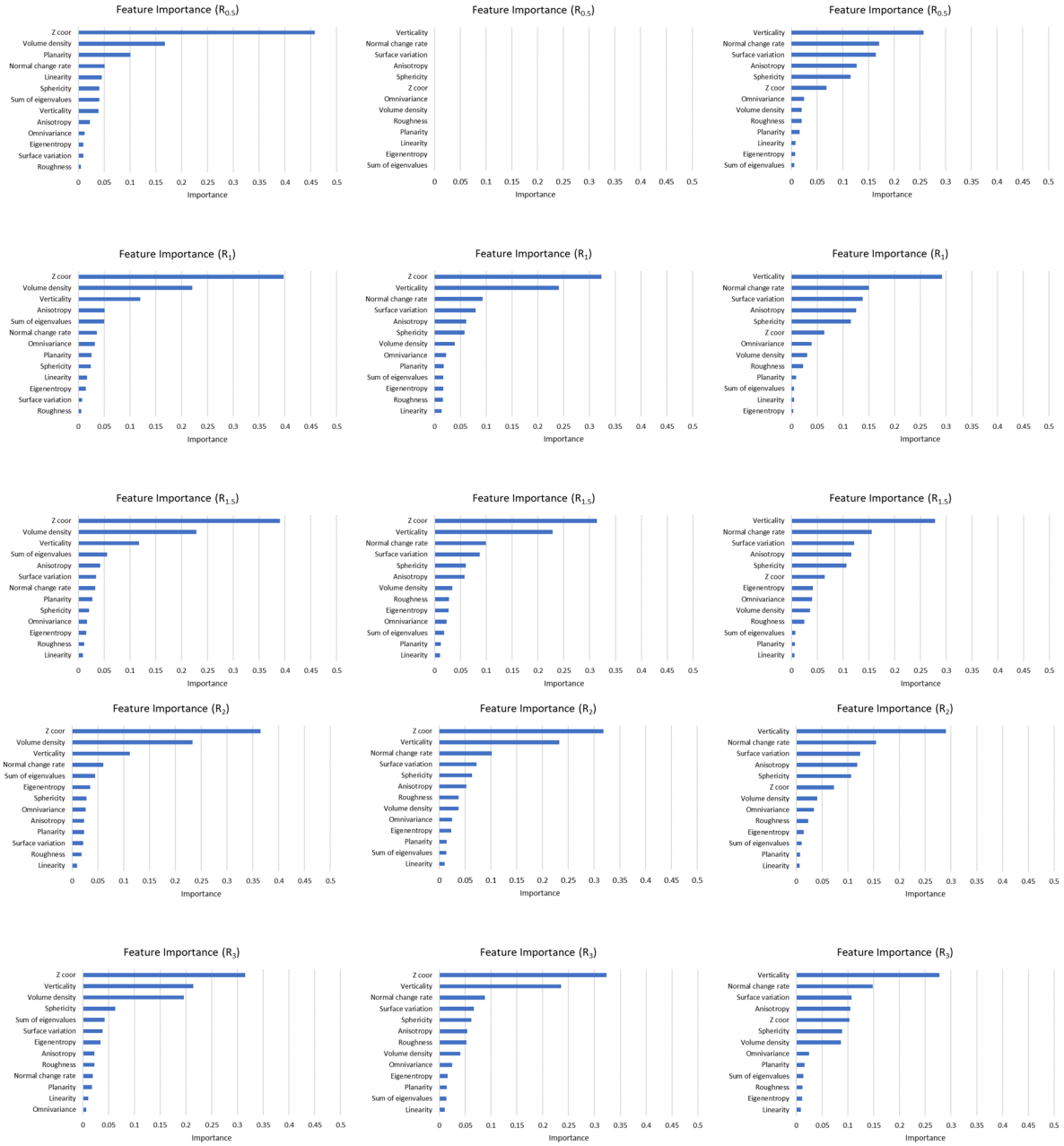

Figure 16. Feature importance rankings. (For Dublin City on the left side; for Vaihingen on the middle; Oakland3D on the right side).

5. Discussion

The results of the study allow general inferences to be made regarding the selection of some optimum parameters for point cloud classification. Determining the optimum support size is one of the most important factors affecting the classification accuracy. Considering the overall accuracy metrics, it was determined that the optimum support

size may not be the same for different classes (Figures 12–15). Furthermore, the idea that the optimum support size depends on the respective point density is meaningful and is consistent with previous studies [9]. In both test sites of the Dublin City dataset which had very high point density, generally, the highest accuracy results were obtained at the R_3 scale, a large support size (Tables 3 and 4). In the Vaihingen and Oakland3D datasets that had less density, generally the highest accuracy results were obtained in R_1 and $R_{1.5}$ support sizes (Tables 5 and 6). Accordingly, it is appropriate to choose a larger support size in denser point clouds and choose a smaller support size in low-density point clouds. Additionally, it has been shown that the optimum support length can vary from one algorithm to another. For classification methods, DT, a rule-based method, generally did not give good results compared to other methods. Although higher classification accuracies are obtained with the instance-based KNN method, it has a long test time. It has been determined that the classification accuracy of LDA, GNB and LR (probabilistic methods), RF (ensemble method), SVM and MLP (neural-network-based method) methods are improved. However, the MLP method has a disadvantage with its long training time. Even though LR, GNB and SVM could compete with other methods in the Dublin City and Vaihingen datasets, they were insufficient in the Oakland3D dataset. The LDA and RF methods also had high classification accuracy in the Dublin City and Oakland 3D datasets, but showed lower performance in the Vaihingen dataset than other classifiers. It was understood that choosing the appropriate classifier depends on the scale and data set. The findings of this study were compared with previous studies with the same datasets. Since the Dublin City dataset is a new dataset, there is no study in the literature comparable to this study. Comparative results for the Vaihingen and Oakland3D datasets are presented in Tables 7 and 8.

Table 7. Comparison with previous studies for the Vaihingen dataset. F1 score values were compared for each class. All values are given in % (WhuY2, ISS_3, K_LDA results were taken from ISPRS website). The highest-obtained accuracies are shown as bold.

Method	Building	Vegetation	Ground	Average F1	Overall Accuracy
This study	74.58	67.75	87.46	76.60	79.71
Özdemir et al. [36]	87.70	71.10	**91.70**	79.40	**84.10**
WhuY2 [37]	**93.10**	**77.30**	88.90	**86.43**	81.00
IIS_3 [37]	83.80	53.20	86.90	74.63	72.50
K_LDA [37]	60.69	64.21	61.05	61.98	50.19

Table 8. Comparison with previous studies for the Oakland3D dataset. F1 score values were compared for each class. All values are given in %. The highest-obtained accuracies are shown as bold.

Method	Building	Vegetation	Ground	Average F1	Overall Accuracy
This study	88.08	93.54	99.48	93.70	**97.30**
Feng and Guo [38]	89.62	86.89	**100.0**	92.17	97.08
Guo and Feng [39]	**94.42**	**94.74**	97.91	**95.62**	96.89
Weinmann et al. [9]	76.98	91.55	98.43	88.99	92.28

Algorithms were also evaluated in terms of test and train time. There are many factors, such as the size of the dataset, the parameters selected and the hardware used, which affect the time. Therefore, although the processing times of the algorithms were specific to this study, they provided important information as they were evaluated under the same conditions.

As expected, algorithms needed more tests and train time for the larger dataset. GNB had the shortest training time, while MLP training time was the longest. MLP as a neural network, having hidden layers, took time for training. According to test times, DT was the fastest algorithm while KNN was the slowest algorithm. KNN was slow because it determined the closest neighbor points for each point individually. In all algorithms except KNN, the training time was longer than the test time. The duration of the algorithms is presented separately for each dataset in Table 9.

Table 9. Training and testing duration of the algorithms (seconds).

Support	LDA	RF	GNB	LR	MLP	DT	SVM	KNN
$Train_{dublin}$	5.140	43.609	1.641	25.100	635.984	19.125	24.359	3.234
$Train_{vaihingen}$	4.250	29.063	1.438	64.141	454.641	15.578	10.361	2.703
$Train_{oakland}$	0.109	4.641	0.016	3.578	98.062	0.265	0.562	0.125
$Test_{dublinA1}$	0.422	8.438	3.906	0.422	4.594	0.250	0.359	169.891
$Test_{dublinA2}$	1.359	17.578	10.281	1.359	14.844	1.141	2.672	833.531
$Test_{vaihingen}$	0.156	4.031	2.360	0.203	1.703	0.125	0.203	149.031
$Test_{oakland}$	0.203	8.953	0.906	0.219	4.984	0.141	0.265	57.390

6. Conclusions

In this study, the performance of eight different machine-learning algorithms for the supervised classification of LiDAR point clouds was investigated. Geometric features of point clouds produced at multi scales were used for classification. Thus, point cloud classification was realized without the need for any additional information other than 3D coordinates. As a result of classification, three classes were selected: building, vegetation and ground surface. The study was repeated in three different areas. In addition, the importance of the geometric features based on the scale and the processing times of the algorithms were examined.

Machine learning methods for point cloud classification have great potential. Appropriate method and parameter selection are important for a correct classification with machine learning. As shown in the results of the study, the appropriate method varies according to the dataset. Due to classified point clouds containing a lot of geometric information, they can be used as a base and reference data for many studies. For future studies, the study can be applied to photogrammetric point clouds that have different characteristics from the LiDAR point cloud.

Author Contributions: Conceptualization, Muhammed Enes Atik, Zaide Duran and Dursun Zafer Seker; Methodology, Muhammed Enes Atik and Zaide Duran; Software, Dursun Zafer Seker; Formal Analysis, Muhammed Enes Atik; Investigation, Muhammed Enes Atik; Resources, Zaide Duran; Data Curation, Muhammed Enes Atik and Zaide Duran; Writing-Original Draft Preparation, Muhammed Enes Atik; Writing-Review & Editing, Zaide Duran and Dursun Zafer Seker; Visualization, Muhammed Enes Atik; Supervision, Zaide Duran. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

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

References

1. Xu, Y.; Ye, Z.; Yao, W.; Huang, R.; Tong, X.; Hoegner, L.; Stilla, U. Classification of LiDAR Point Clouds Using Supervoxel-Based Detrended Feature and Perception-Weighted Graphical Model. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *13*, 72–88. [CrossRef]

2. Duran, Z.; Aydar, U. Digital modeling of world's first known length reference unit: The Nippur cubit rod. *J. Cult. Herit.* **2012**, *13*, 352–356. [CrossRef]
3. Lin, C.H.; Chen, J.Y.; Su, P.L.; Chen, C.H. Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification. *ISPRS J. Photogramm. Remote Sens.* **2014**, *94*, 70–79. [CrossRef]
4. Guo, B.; Huang, X.; Zhang, F.; Sohn, G. Classification of airborne laser scanning data using JointBoost. *ISPRS J. Photogramm. Remote Sens.* **2015**, *100*, 71–83. [CrossRef]
5. Thomas, H.; Goulette, F.; Deschaud, J.E.; Marcotegui, B.; LeGall, Y. Semantic classification of 3D point clouds with multiscale spherical neighborhoods. In Proceedings of the 2018 International conference on 3D vision (3DV), Verona, Italy, 5–8 September 2018; pp. 390–398.
6. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Contextual classification of lidar data and building object detection in urban areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 152–165. [CrossRef]
7. Zolanvari, S.M.; Ruano, S.; Rana, A.; Cummins, A.; da Silva, R.E.; Rahbar, M.; Smolic, A. DublinCity: Annotated LiDAR Point Cloud and its Applications. In Proceedings of the 30th BMVC, Cardiff, UK, 9–12 September 2019.
8. Munoz, D.; Bagnell, J.A.; Vandapel, N.; Hebert, M. Contextual classification with functional max-margin markov networks. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 975–982.
9. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *Isprs J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [CrossRef]
10. Vosselman, G.; Coenen, M.; Rottensteiner, F. Contextual segment-based classification of airborne laser scanner data. *ISPRS J. Photogramm. Remote Sens.* **2017**, *128*, 354–371. [CrossRef]
11. Mayr, A.; Rutzinger, M.; Bremer, M.; Oude Elberink, S.; Stumpf, F.; Geitner, C. Object-based classification of terrestrial laser scanning point clouds for landslide monitoring. *Photogramm. Rec.* **2017**, *32*, 377–397. [CrossRef]
12. Belgiu, M.; Tomljenovic, I.; Lampoltshammer, T.J.; Blaschke, T.; Höfle, B. Ontology-based classification of building types detected from airborne laser scanning data. *Remote Sens.* **2014**, *6*, 1347–1366. [CrossRef]
13. Li, X.; Cheng, X.; Chen, W.; Chen, G.; Liu, S. Identification of forested landslides using Lidar data, object-based image analysis, and machine learning algorithms. *Remote Sens.* **2015**, *7*, 9705–9726. [CrossRef]
14. Plaza-Leiva, V.; Gomez-Ruiz, J.A.; Mandow, A.; García-Cerezo, A. Voxel-based neighborhood for spatial shape pattern classification of lidar point clouds with supervised learning. *Sensors* **2017**, *17*, 594. [CrossRef]
15. Cabo, C.; Ordóñez, C.; Sáchez-Lasheras, F.; Roca-Pardiñas, J.; de Cos-Juez, J. Multiscale Supervised Classification of Point Clouds with Urban and Forest Applications. *Sensors* **2019**, *19*, 4523. [CrossRef]
16. Becker, C.; Rosinskaya, E.; Häni, N.; d'Angelo, E.; Strecha, C. Classification of aerial photogrammetric 3D point clouds. *Photogramm. Eng. Remote Sens.* **2018**, *84*, 287–295. [CrossRef]
17. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
18. Hastie, T.; Tibshirani, R.; Friedman, J. Random Forests. In *The Elements of Statistical Learning*, 2th ed.; Springer: New York, NY, USA, 2010; pp. 587–604.
19. Duda, R.O.; Hart, P.E. *Pattern Classification and Scene Analysis*; Wiley: New York, NY, USA, 1973; Volume 3, pp. 731–739.
20. Domingos, P.; Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Mach. Learn.* **1997**, *29*, 103–130. [CrossRef]
21. Dey, L.; Chakraborty, S.; Biswas, A.; Bose, B.; Tiwari, S. Sentiment analysis of review datasets using naive bayes and k-nn classifier. *Int. J. Inf. Eng. Electr. Bus* **2016**, *8*, 54–62. [CrossRef]
22. Lou, W.; Wang, X.; Chen, F.; Chen, Y.; Jiang, B.; Zhang, H. Sequence based prediction of DNA-binding proteins based on hybrid feature selection using random forest and Gaussian naive Bayes. *PLoS ONE* **2014**, *9*, e86703. [CrossRef]
23. Alpaydin, E. *Introduction to Machine Learning*, 2th ed.; MIT Press: London, UK, 2010.
24. Abdullah, M.H.A.; Othman, M.; Kasim, S.; Saharuddin, S.S.; Mohamed, S.A. A Spiking Neural Networks Model with Fuzzy-Weighted k-Nearest Neighbour Classifier for Real-World Flood Risk Assessment. In *International Conference on Soft Computing and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 222–230.
25. Rymarczyk, T.; Kozłowski, E.; Kłosowski, G.; Niderla, K. Logistic Regression for Machine Learning in Process Tomography. *Sensors* **2019**, *19*, 3400. [CrossRef]
26. Cox, D.R. The regression analysis of binary sequences. *J. Royal Stat. Soc. Ser. B* **1958**, *20*, 215–232. [CrossRef]
27. Peng, C.Y.J.; Lee, K.L.; Ingersoll, G.M. An introduction to logistic regression analysis and reporting. *J. Educ. Res.* **2002**, *96*, 3–14. [CrossRef]
28. Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **1936**, *7*, 179–188. [CrossRef]
29. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. Machine Learning Basics. In *Deep Learning*; MIT press: Cambridge, UK, 2016; Volume 1, pp. 99–166.
30. Cortes, C.; Vapnik, V. "Support-vector networks". *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
31. Lei, Y. Individual intelligent method-based fault diagnosis. In *Intelligent Fault Diagnosis and Remaining Useful Life Prediction of Rotating Machinery*; Butterworth-Heinemann: Oxford, UK, 2017; pp. 67–174.
32. Chakroun, I.; Haber, T.; Ashby, T.J. SW-SGD: The sliding window stochastic gradient descent algorithm. *Procedia Comput. Sci.* **2017**, *108*, 2318–2322. [CrossRef]

33. Sharma, A. Guided stochastic gradient descent algorithm for inconsistent datasets. *Appl. Soft Comput.* **2018**, *73*, 1068–1080. [CrossRef]
34. Zhang, N.; Zhu, J. Privacy-preserving access control scheme for outsourced data in cloud. In *Workshop on E-Business*; Springer: Cham, Switzerland, 2016; pp. 215–224.
35. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Int. J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
36. Özdemir, E.; Remondino, F.; Golkar, A. Aerial point cloud classification with deep learning and machine learning algorithms. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 843–849. [CrossRef]
37. ISPRS Semantic Labeling Contest (3D): Results. Available online: https://www2.isprs.org/commissions/comm2/wg4/results/vaihingen-3d-semantic-labeling/ (accessed on 12 February 2021).
38. Feng, C.C.; Guo, Z. Automating parameter learning for classifying terrestrial LiDAR point cloud using 2D land cover maps. *Remote Sens.* **2018**, *10*, 1192. [CrossRef]
39. Guo, Z.; Feng, C.C. Using multi-scale and hierarchical deep convolutional features for 3D semantic classification of TLS point clouds. *Int. J. Geogr. Inf. Sci.* **2020**, *34*, 661–680. [CrossRef]