University of Oslo

Project 3

The Use of Machine Learning and LiDAR for Urban Tree Species Classification

FYS-STK4155

José Luis Barreiro Tomé

December 2022

Contents

Abstract:

Trees are important components in cities. Their functions go from improving air quality and CO_2 mitigation to leisure and temperature dampening due to Urban Heat Island effect. Open-source urban tree datasets are becoming more often available and can be used to monitor vegetation in cities. Light Detection and Ranging (LiDAR) is a remote sensing method commonly used to extract structural information in vegetation and other objects. It is also often freely available and used in multiple applications. The aim of this study is to use open-source data to test tree species classification in the city of Washington DC, USA. The performance of several Machine Learning classification methods is compared. Preprocessing of the LiDAR data, tree crown segmentation, feature extraction, tree species classification and analysis performance are the main steps of the study. The overall accuracies were 74.48%, 76.35%, 75.52%, 74.69% and 72.20% for the SVM, RF, SVM, logreg and DT classifiers, yet the performance was significantly better for deciduous than for coniferous species.

1. Introduction

Over 55% of the total population lives nowadays in urban areas, a proportion that is expected to increase to 68% by 2050 (United Nations, 2019). Urban areas are characterized by a large fraction of impervious surfaces, like concrete or asphalt. Absorption and reflection of solar radiation is altered, making temperatures over urban areas higher than those of neighboring rural areas. This is the well-known Urban Heat Island effect (Oke 1982). Green infrastructures and vegetation areas help decreasing the urban temperature, providing an essential effect. In addition, they mitigate carbon dioxide (CO_2), reduce urban floods, wind speed, noise pollution and support biodiversity in the urban areas. On the other hand, allergic reactions can be named as adverse effects on the urban population. Different tree species require different maintenance and have different benefits and disadvantages for the local population. For all those reasons, a correct monitoring of urban tree species is necessary for urban planning.

The traditional methods of monitoring urban tree species include costly field surveys or photointerpretation. These methods can be successful in a small scale, but large cities need better tools for having a correct overview. In the last years, remote sensing is becoming an affordable way of monitoring the tree canopy cover, although many challenges remain to keep track of the state of individual trees. Some authorities spend some resources to create and update databases with city trees. In other cases, there is collaborative crowdsourcing initiatives, such as *OpenTreeMaps* (https://www.opentreemap.org), who help with the tree inventory.

More and more platforms are becoming available for capturing remote sensing data capable of providing necessary information for monitoring vegetation in cities. Aerial surveys, satellite images, terrestrial scanning or drones can make this function, either capturing spectral information (images) or other kinds. LiDAR (acronym of Light Detection And Ranging) is a remote sensing method that uses light in the form of a pulse laser to measure ranges (or distances) to objects, being able to calculate object position. Airborne Laser Scanning (ALS hereafter) campaigns over extended areas, subsidized by local authorities, are becoming available as open-source products, in an increasing trend. The result of a survey is a dense group of elevation points. These so-called *point cloud*s provide an estimate of three-dimensional forest or tree structure, understory vegetation and terrain. In addition to the spatial information, LiDAR intensity is recorded as the return strength of each laser beam. Low intensity implies low reflectivity and high intensity indicates high reflectivity. With this information is possible, for example, to separate objects by its reflectivity.

Machine Learning algorithms are used in many remote-sensing applications, some of them regarding vegetation analysis. There are many classifiers available, some of which can adapt very well to LiDAR-based tree species classification. MultiLayer Perceptron, Random Forest (RF hereafter), Logistic Regression (logreg), Support Vector Machine (SVM), K-Nearest Neighbor… are some examples. Simpler models for classification can perform well in many cases. But their applicability is limited when the target behaves in a non-linear way. By expanding to non-linear and high dimensional cost function optimization problem, the necessity of iterative solvers becomes evident as the computational cost of matrix inversion becomes unbearable.

Several studies have tested the use of LiDAR-based features to classify tree species using machine learning methods. They distinguished either coniferous/deciduous or a larger number of different tree species. Ørka et al. (2009) classified coniferous and deciduous species using a linear discriminant analysis (LDA) in boreal forest conditions in Norway using intensity and ALS-derived structured features, with an overall classification accuracy of 88%. Cetin and Yastikli (2022) used three different classification algorithms using intensity- and spatial-based LiDAR features in a small urban area in Turkey, with overall accuracies of 80%, 83.75 % and 73.75% for SVM, RF and MLP classifiers, respectively.

The motivation of this study is to assess the potential of different classification algorithms for classification of trees in a large city: Washington DC in USA, using open-source material as input in the models: a LiDAR dataset collected in 2018 and an open-source urban tree dataset maintain by the District of Columbia. The study from Cetin and Yastikli (2022) is taken as a guideline, although some different methods for ALS preprocessing and tree segmentation are proposed. Some simpler classification algorithms (logreg and simple DT) are added to the analysis.

The ALS data are preprocessed as a preliminary step. Normalized height from ground is computed and a Canopy Height Model (CHM) is produced, as an input for tree segmentation. Here an inverse watershed algorithm computes the bi-dimensional shape of individual tree crowns. These polygons are used to isolate the LiDAR points per tree, to extract spatial and intensity predictors. This information is crossed with the species-labeled points from the urban tree dataset, to be used as features in the classification models. Different algorithms are tested in the classification phase, from simpler logreg and DT, to MLP and ensembled methods, RF and SVM.

2. Data and methods

2.1. Theory

2.1.1. Neural Network

Neural Networks, often called Multi-Layer Perceptron (MLP), are nonlinear, statistical models, that can learn to approximate some regression function or classification model. They are designed to mimic a human brain, in which an arbitrary number of neurons within a layer send signals to other neurons in a different layer. Each neuron accumulates its incoming signals, and they must exceed an activation threshold to yield an output. Each connection is represented by a weight (w_i) variable plus some bias. They are called feedforward when the information flows just in one direction through the model, from the input, through the layers and produces an output. They are called networks because they can be composed by many different functions. For classification problems, there is often an output function on the output layer. For regression there is typically only one output, and the output function would often be the identity function.

The performance of the neural network is as follows. Each input x_i is multiplied by a weight w_i, that indicate the strength of the connection to each node in the next layer. To avoid outputs of only zeros, a small bias (b_i) is added to each weight. These weights are usually initialized with random small numbers distributed around zero. The result is analyzed by an activation function (zh), and the result is the output for each node, which at the same time becomes the input for the next layer.

$$y = f(\sum_{i=1}^{n} w_i x_i + b_i) = f(z) \tag{1}$$

The training data shows the behavior of the output layer: the output data should be close to y. But it doesn't specify the behavior of the hidden layers, the learning algorithm decides how to use them to end producing the desired output (that is the reason for calling them hidden layers).

There are several choices of activation functions. In this project ReLU was used. ReLU works well in deep neural networks and it's quite fast to compute This function only activates when the input variable that goes through it is positive:

$$ReLU = \begin{cases} 0 & if\ x < 0 \\ x & if\ x \geq 0 \end{cases} \tag{2}$$

Backpropagation algorithm: After one feed-forward pass through the network (i.e., one iteration), we need to adjust the weights and biases that connect the neurons in the network, to minimize the errors in the output. After performing the feed-forward, the output error (last layer L) is given by

$$\delta_j^L = f'(z_j^L) \frac{\partial C}{\partial(a_j^L)} \tag{3}$$

where $f'(zjL)$ is the derivative of the activation function of the last j neuron, ajL is the output data of the last j neuron and C is the cost function.

The back-propagate error can be computed for each layer l=L−1,L−2,…,2 as

$$\delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} f'(z_j^l)$$

Weights and biases can now be updated using GD for each l=L−1,L−2,…,2 according to

$$w_{jk}^l \leftarrow= w_{jk}^l - \eta \delta_j^l a_k^{l-1},$$

$$b_j^l \leftarrow b_j^l - \eta \frac{\partial C}{\partial b_j^l} = b_j^l - \eta \delta_j^l \tag{5}$$

where η is the learning rate

2.1.2. Logistic regression

Logistic Regression (logreg) is often used to estimate the probability that an instance belongs to a particular class. Depending on the probability being under or equal/over 50%, the model decides that the instance belongs to one or another class. To do so, the model computes a weighted sum of the input features (plus a bias term) and output the *logistic* of the results. The *logistic* is a sigmoid function that outputs a number between 0 and 1:

$$\sigma(t) = \frac{1}{1+\exp(-t)} \qquad (6)$$

2.1.3. Decision trees

DT are supervised machine learning algorithms that can be used for both classification and regression operations. They are also the basis of other powerful algorithms, from random forest to extreme gradient boosting (XGBoost).

The algorithms try to find the descriptive features which contains the most information regarding the target feature. The decision tree is divided into a root node, interior nodes and the final leaf nodes, all connected by branches. The leaf nodes contain the predictions, or classification of a given instance. The classification process starts at the root node, finding the most informative feature, and moving down the tree branch, testing the attribute specified by each node, until a stopping-criteria ends up in the leaves.

Scikit-Learn uses the *Classification And Regression Tree* (CART) algorithm to train DT (Geron, 2019). The algorithm splits the training set into two subsets (binary) using a feature *k* and one threshold. The way *CART* algorithm choses the feature and weight is by searching for the purest subsets (weighted by their size). The cost-function that the algorithm tries to minimize is:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \qquad (7)$$

where G measures the impurity of the subset and m is the number of instances in the subset.

Once the dataset has been split in two, the sub-sets are also split recursively, until it reaches the maximum depth of the tree (defined as a hyperparameter).

Decision trees can use different criteria for splitting a node, being the most common the *Gini index* and the *entropy (eq. 9)*. The *Gini index* classifies observations to classes k_i with probability p_{mk} (Hastie and Friedman, 2019). If we code each observation as 1 for class *k* and zero otherwise, the sum over classes for the variance over the node of this 0-1 response is:

$$g = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \qquad (8)$$

The concept of entropy is originated in thermodynamics as a measure of molecular disorder (Geron, 2019). In Machine Learning, entropy approaches zero when it contains instances of only one class. Equation 9 shows the definition of the entropy of the ith node:

$$s = -\sum_{k=1}^{K} p_{mk} \log p_{mk} \qquad (9)$$

2.1.4. Random Forest

RF is an ensemble of Decision Tress, generally trained via the bagging method. It has all the hyperparameters of a DT classifier, plus all the parameters of a bagging classifier.

The RF provides a variable importance measure, to identify the most relevant features, which is an interesting feature, not available in other Machine Learning algorithms. It looks at how much the tree nodes that use a particular feature reduce impurity on average.

2.1.5. Support Vector Machines

Support Vector Machines (SVM hereafter) is a non-parametric supervised machine learning algorithm, capable of performing classification or regression (Hastie and Friedman, 2019). It is well-suited for classification of complex but small-sized datasets (Hjort-Jensen, 2022). The theory behind SVM is based on the mathematical description of the hyperplanes. A hyperplane is a decision boundary that differentiates the two domains, or classes in SVM. The support vectors are the data points closest to the hyperplane. If $\beta\_0, \beta\_1, \dots, \beta_p \in \mathbb{R}$, the hyperplane *H* in $\mathbb{R}^p$ is:

$$H = \{(z_1\ z_2 \dots\ z_p) |\ \beta_0 + z_1\beta_1 + z_2\beta_2 + \cdots + z_p\beta_p = 0\} \qquad (10)$$

assuming that $\sum_k \beta_k^2 = 1$

A given dataset is linearly separable if there exist a hyperplane *H* in $\mathbb{R}^p$ that satisfy:

$$t_i(\beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ip}\beta_p) > 0 \qquad (11)$$

SVM tries to find the best possible hyperplane, meaning the one with the largest margin between the two classes, so-called margin hyperplane. That means to chose $\beta_0, \beta_1 \dots ., \beta_p$ such that $M > 0$ is maximized under

$$t_i(\beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ip}\beta_p) > M \qquad (12)$$

for all data points x_i

But data is not always linearly separable. In order to get usable classifications, a slack variable $\varepsilon_i \geq 0$ is introduced, accepting hyperplanes that do not separate the points totally clean. The new objective is to maximize $M > 0$ under the conditions

$$t_i(\beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ip}\beta_p) > M(1 - \varepsilon_i) \qquad (13)$$

where $\sum \varepsilon_i \leq C$, being C a hyperparameter that sets which points are allowed to be misclassified.

A better class separation in datasets not easy to classify can be achieved by *kernel* methods, which are complex functions that change the basis from $x \to z = \emptyset(x)$. Some of the available *kernels* in scikit learn are linear, polynomial, Gaussian Radial Basis Function (RBF) or tanh. In this project the RBF function will be used:

$$K(x,y) = \exp\left(-\gamma\|x-y\|^2\right) \tag{14}$$

2.2. Study area and data

The LiDAR point-cloud data is an open dataset and can be download from https://opendata.dc.gov/apps/lidar-grid-tiles/explore, tiled and classified as LAS 1.4 format (ASPRS, 2013). The data was captured in a single flight on April 5, 2018, at a flying altitude of 3000 m. A Piper Navajo PA31-50 plane was used, and the LiDAR sensor was the Leica ALS80. It covers the whole District of Columbia, at an average point density of 4.5 points per square meter. A series of automatic filtering, classification and manual editing was conducted by the contractor to set all points in different classes, such as Ground, low vegetation, medium vegetation, high vegetation, buildings, water etc.

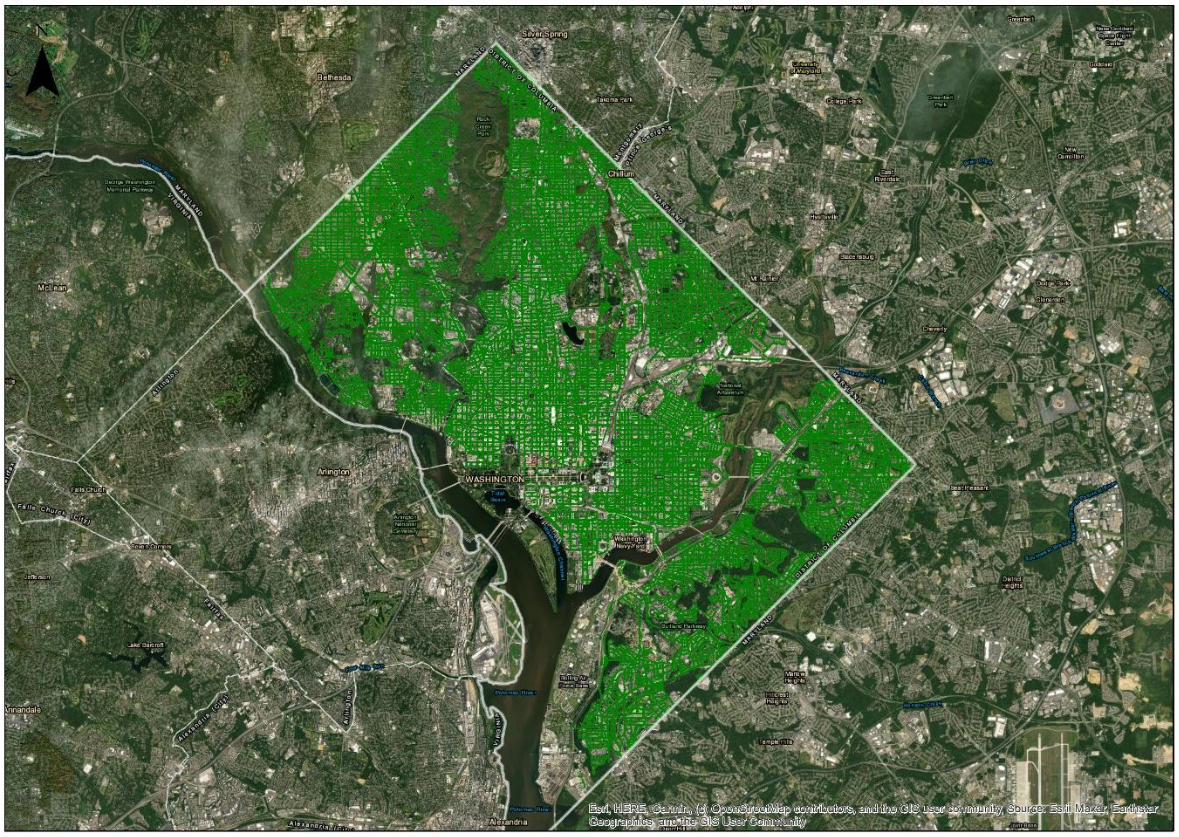
WASHINGTON
Arlington
Silver Spring
Bethesda
Alexandria
Esri, HERE, Garmin, (c) OpenStreetMap contributors, and the GIS user community, Source: Esri, Maxar, Earthstar Geographics, and the GIS User Community

Figure 1. *Map of study area with Washington DC Urban Forestry Street Trees overlay*

Ground-truth data for tree species came from the Washington DC Urban Forestry Street Trees, also an open dataset used for management and maintenance of street trees on the District of Columbia. The dataset contains locations and attributes of trees, such as tree species. It covers the whole city district, including a majority of deciduous trees, from majestic willow oaks to more common American beech. Just around 5% of the trees registered are conifer trees, like bald cypress or pine trees. The dataset can be downloaded in different formats from https://opendata.dc.gov/datasets/DCGIS::urban-forestry-street-trees/about. There are registered more than 200 000 trees.

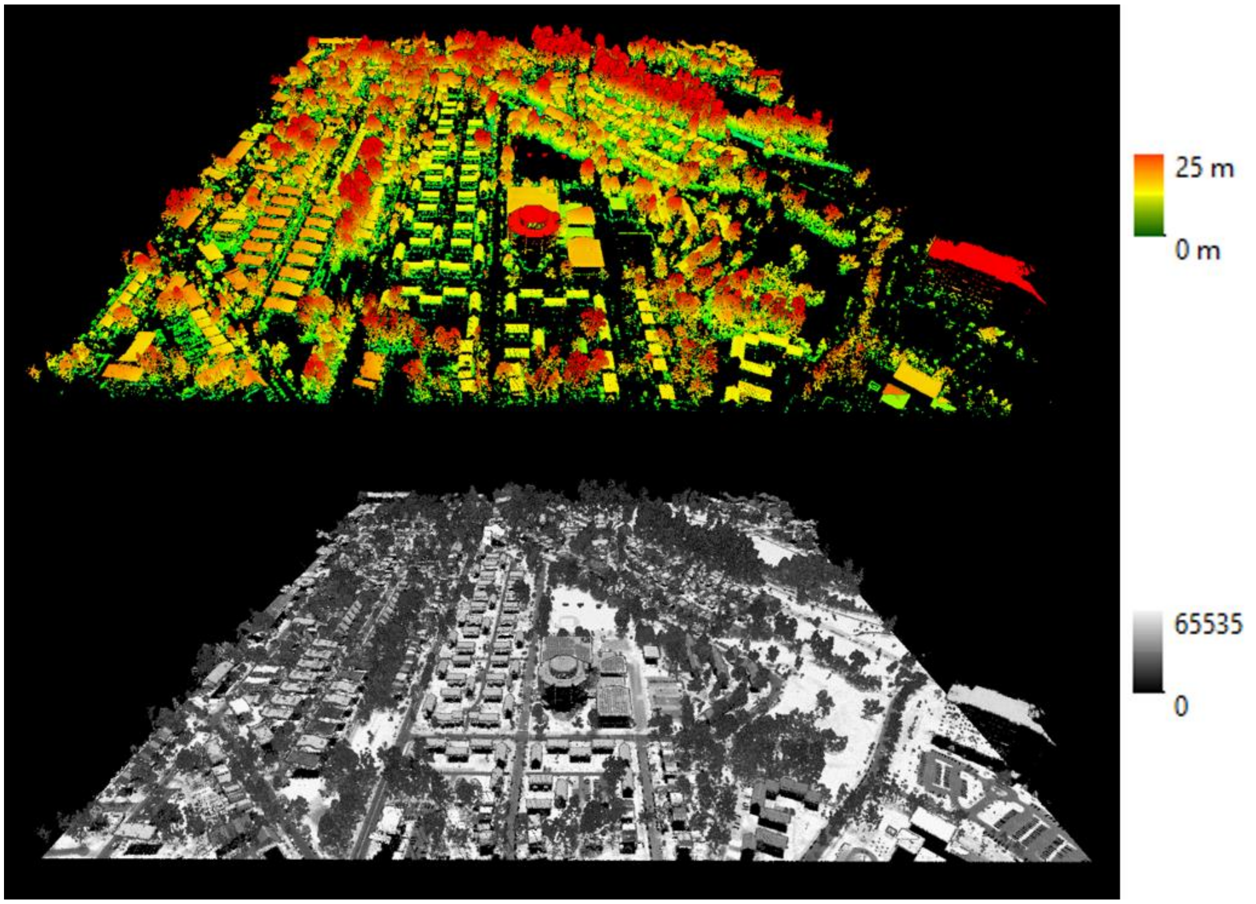

***Figure 2**. Example of LiDAR 3D point cloud, with color representing heights (top) and intensity (bottom)*

2.3. General workflow

The main steps taken in this work for tree species classification of urban trees using 3D LiDAR data are visualized in Figure 3 and explained in more detail thereafter. The software used for all pre-processing of the ALS data and feature extraction was *LaStools* (Isenburg, 2022). *FWTools* and *GDAL* (Rouault et al. 2022) were used for mosaicking the raster tiles into one. Tree crown segmentation was performed with *eCognition Developer 10.2* (Trimble, 2021), and validation of predicted tree crowns was performed in QGIS 3.28 (QGIS, 2022). The Python 3.8 programming language was used to implement the classification algorithms, for which functionalities of scikit-learn (Pedregosa *et al.* 2011) package were used. Other packages used in the implementation were numpy (Harris et al. 2020), pandas (McKinney, 2010) and seaborn (Waskom, 2020).

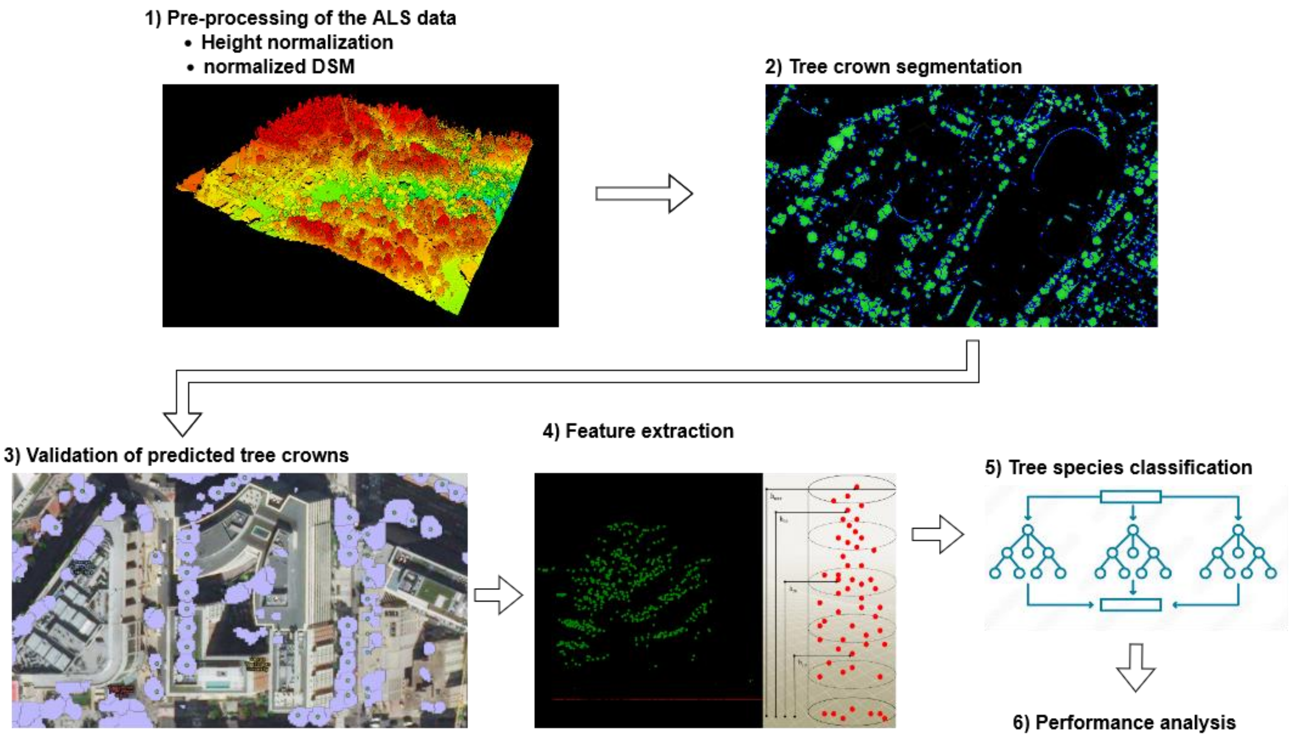
1) Pre-processing of the ALS data
• Height normalization
• normalized DSM
2) Tree crown segmentation
3) Validation of predicted tree crowns
4) Feature extraction
5) Tree species classification
6) Performance analysis

Figure 3. *Flowchart representing general workflow of this study*

2.3.1. Pre-processing of the ALS data

The first step for preparing the ALS dataset were normalize all 3D LiDAR points, i.e. it was computed the height above ground of each point with respect to the *Ground* class points. This was done by constructing a ground TIN as a ground model, per LiDAR tile. A buffer of 50 m around the tile was considered when computing the points of each tile to avoid boundary-effects at the edges, thereafter the tile is exported to its original boundaries. Once this new dataset was obtained, a CHM elevation grid was produced in raster format, with 50x50 centimeters raster cell (or pixel). This raster constituted the normalized Digital Surface Model (nDSM).

2.3.2. Tree Crown segmentation

nDSM raster from previous step was used to compute image segmentation with the *inverse watershed* algorithm. This is a region-based segmentation method, where an elevation raster is treated as a topographic landscape with ridges and valleys. The algorithm divides the image into *catchment* basin. The basin includes all pixels whose path of steepest descent terminates at its minimum. In the case of tree segmentation, the *inverse* segmentation treats the treetop as the *local minimum*, and the basin as the tree canopy. The result dataset is a geospatial vector data format with polygons representing individual trees or other objects over the whole project area.

2.3.3. Validation of predicted tree crowns

Information from the ground-truth dataset and the tree-crown segmentation were crossed in this step. A spatial join between the tree canopy polygon and the tree species dataset (represented as a point with coordinates x and y) was performed. Each polygon acquired the tree species information from the point falling inside its domain. Because of inaccuracies in the single tree segmentation and in the position of trees on the ground-truth dataset, some polygons might include more than one point. Only polygons with just one point were selected. Besides, a visual inspection of all polygons against the nDSM was performed to filter out more errors. The result dataset from this step is a geospatial vector data format with polygons representing individual trees, properly classified with respect to tree species.

2.3.4. Feature extraction

The features used as input in the machine learning-based classification algorithms for the classification of urban tree species were calculated using the height and intensity information of the LiDAR points. As suggested by Cetin and Yastikli (2022), all LiDAR and intensity variables exposed in Table 1 were calculated per tree canopy polygon. Only the points classified as *High Vegetation* from the normalized height from ground dataset were used. These features were joined back to the original dataset with tree species information, making ready the input dataset for the classification.

***Table 1**. The generated spatial- and intensity-based features from LiDAR data*

LiDAR Data	
Spatial-Based Features	**Intensty-Based Features**
Number of points	-
Maximum Z	Maximum intensity
Minimum Z	Minimum intensity
Standard deviation of Z	Standard deviation of intensity
Mean Z	Mean intensity
Skewness of Z	Skewness of intensity
Kurtosis of Z	Kurtosis of intensity
Z range	Intensity range
5th percentile of Z	5th percentile of intensity
25th percentile of Z	25th percentile of intensity
50th percentile of Z	50th percentile of intensity
75th percentile of Z	75th percentile of intensity
90th percentile of Z	90th percentile of intensity

2.3.5. Tree species classification

Before performing classification, class labels were assigned as coniferous (0) or deciduous (1) species. Feed-forward Neural Network (or MLP), DT, RF and SVM were the chosen machine learning classifiers to be compared.

The choice of architecture, activation function, learning rate and regularization parameter tuning could have a great impact on the performance of the MLP for classification. A grid-search was performed to decide which values of learning rate and regularization parameter show higher train- and test- accuracy score (Figure 4). In addition, this operation was performed using different activation functions. Results show that they are not very sensitive to the choice of regularization parameter and learning rate. A regularization parameter of 1.e-2 and a learning rate of 1.e-7, when using a ReLU activation function with *lbfgs* solver showed marginally better results.

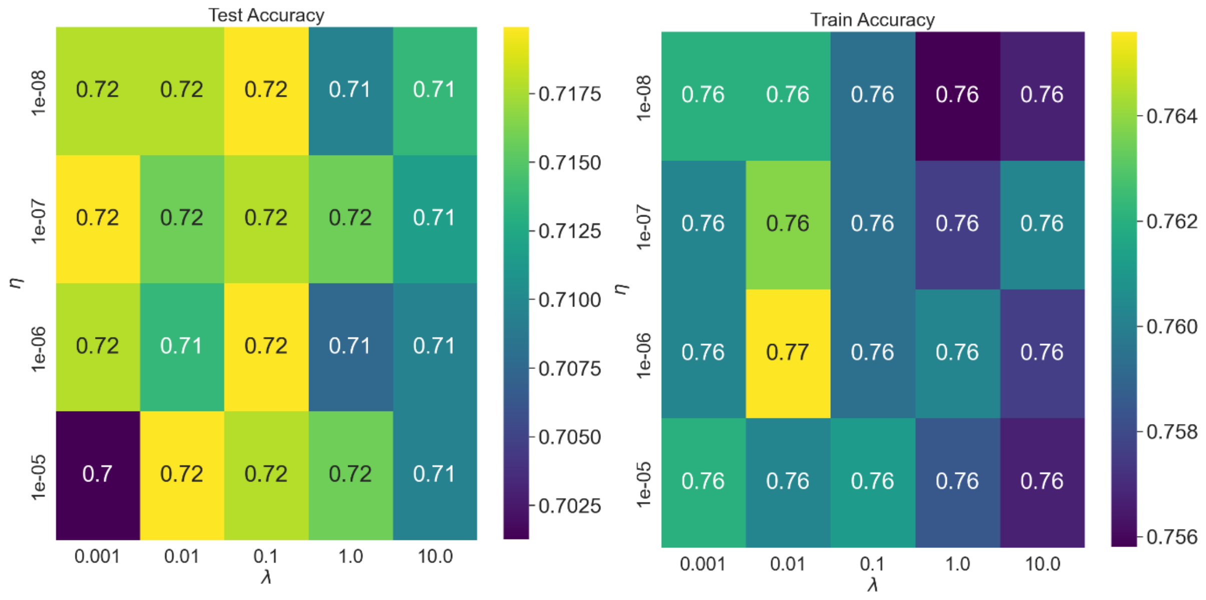

Figure 4. *Grid search plots for urban tree classification task with learning rate and regulatization as hyperparameters using a MLP with a single layer and 1 node, ReLU as activation function and lbfgs as solver*

A bootstrap resampling method was used to find the number of nodes per layer and number of layers that showed a smaller mean squared error when performing the MLP classification. Figure 5 shows the result of the test. Just 1 node in 1 hidden layer was chosen as best model complexity for the MLP model.

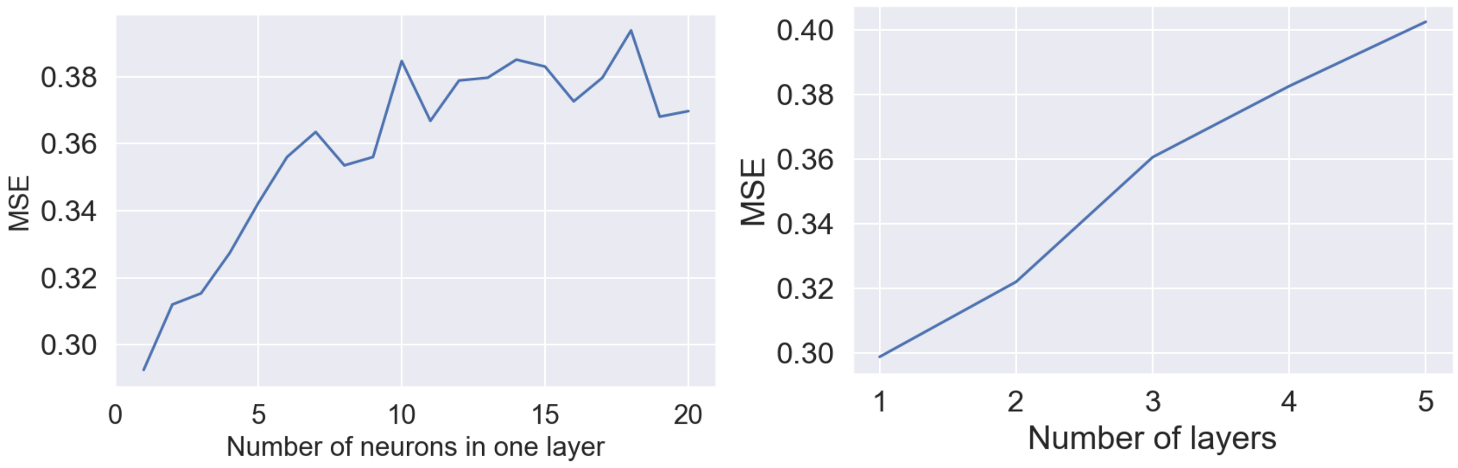

Figure 5. Test for number of nodes and number of layers against mean squared error in MLP model for urban tree classification

For logistic regression, again a grid search was used to find the best regularization strength parameter. Figure 6 shows accuracy scores for different regularization values. The differences were again marginal, but a regularization parameter of 0.1 was chosen. The default L2 penalty and *lbfgs* solver were used.

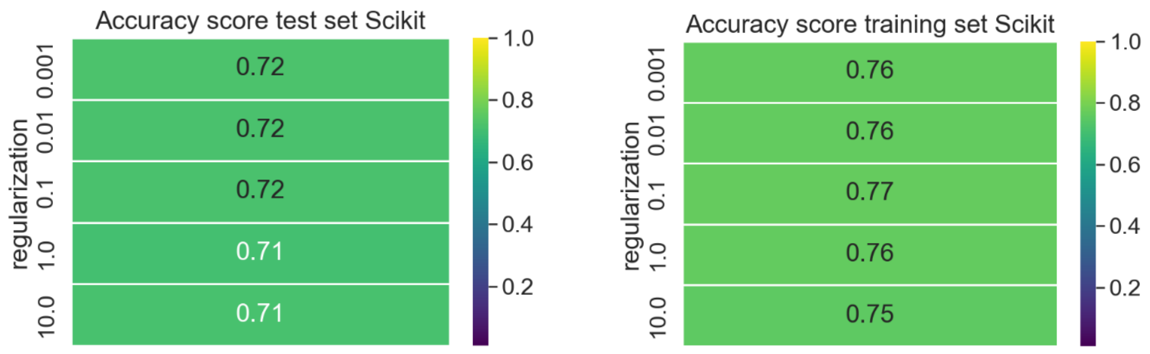

Figure 6. *Grid search plot for urban tree classification task depending on regularization parameter for logistic regression*

Figure 7 shows a heatmap with different score values for Random Forest classifier depending on the number and depth of the trees. The number of trees (n_estimator in scikit learn functionality) and the number of splits that each decision tree is allowed to make was tested. The performance acuracy of the number of trees is stabilizing after approximately 30-40 estimators. To add more weak tree estimators will not add any value in the prediction, then the number of trees chosen is 40. Regarding the depth of the tree, a too low value would produce underfitting, and a too high value will overfit the data. To find the right balance, accuracy score was compared for train and test data. After 4 splits the training score is increasing over the test score. In this case it means that from this point the model cannot predict equally well when using unseen new data, producing overfitting. Therefore, a tree depth value of 4 splits was chosen. Similar results were found for *Gini* or *Entropy* as the function to measure the quality of a split. The default *Gini* was chosen. The same tree depth and function were used when testing the simple decision tree classifier.

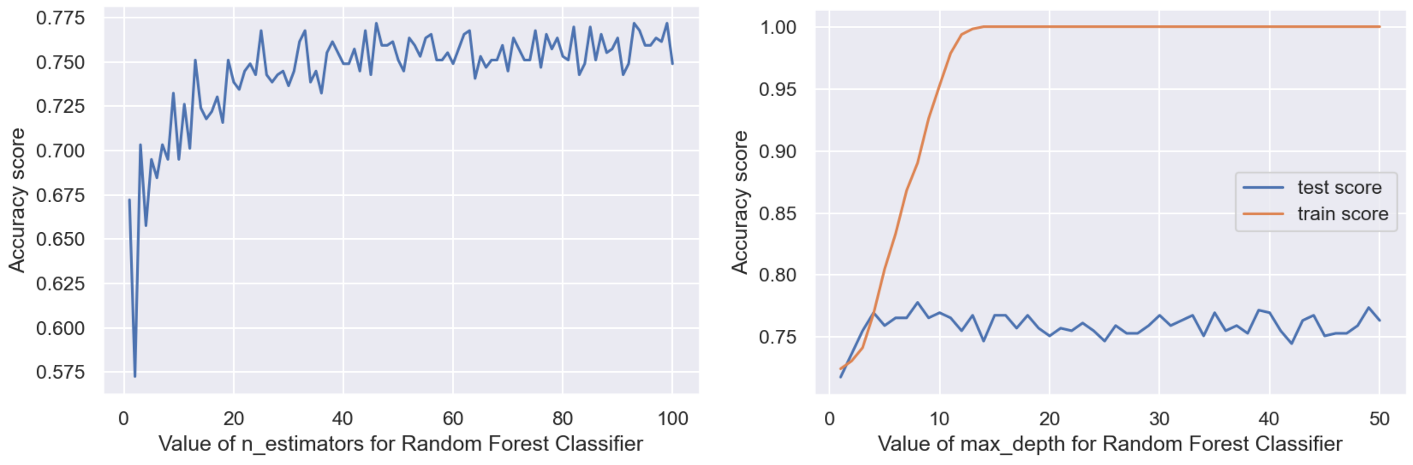

Figure 7. *Test for number and depth of trees on Random Forest classifier for urban tree classification*

Regarding SVM, different regularization parameters

When using the SVM classifier in scikit learn, it was done an hyperparameter tuning to get the best performance. Figure 8 shows the accuracy score for SVM classifier as function of the regularization parameter C. The strength of the regularization is inversely proportional to C. The accuracy score remains fairly constant through the test, but around C=20 the accuracy score has highest value. The gamma parameter is the kernel coefficient for *kbf* (kernel based filter), which is used in this study. The optimal parameter to maximize accuracy but avoid overfitting is 1.e-8.

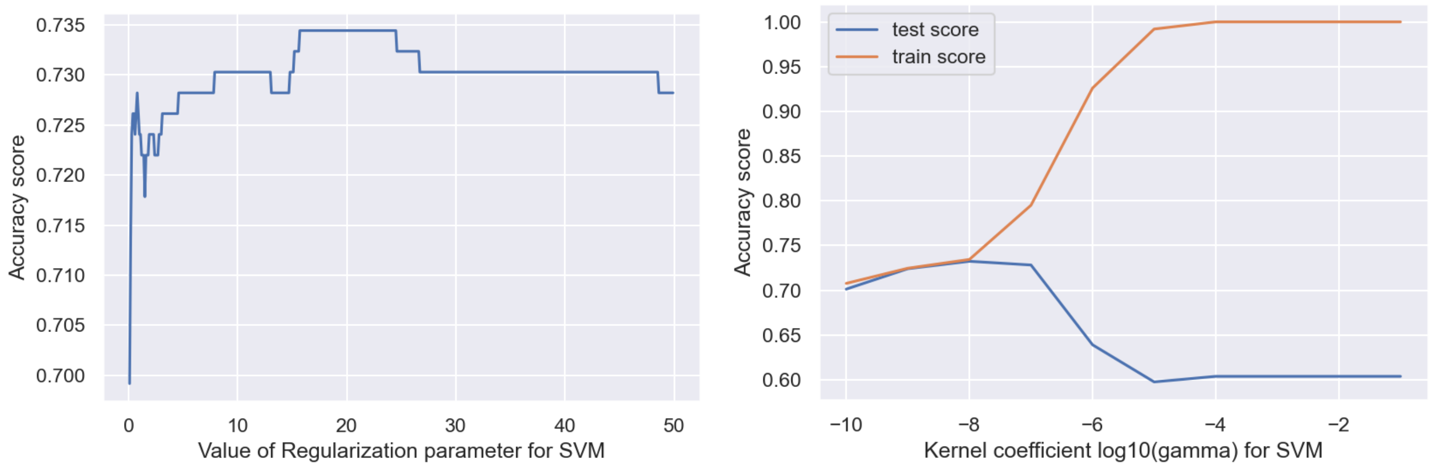

Figure 8. *Test for regularization parameter C and kernel coefficient δ on SVM for urban tree classification*

2.3.6. <u>Performance analysis</u>

Scaling the input data is an important step in data preparation for achieving higher performance, making an equal contribution of each feature. The standard scaling was used on the model features. Data was transformed to be normally distributed within each feature and scaled such that the mean is zero and the standard deviation is 1.

Cross-validation techniques made possible to validate the stability of all proposed classification methods. The samples are divided into equally sized exhaustive and mutually exclusive subsets. At each split (or fold), one of these subsets is used as test set while the rest are used together as training set. The process is repeated for all possible training and test sets. A performance accuracy value is acquired as the average of each split result.

The data was divided into training (70%) and testing (30%) set for performing the classification. Accuracy, recall, precision and F1-score for tree species classification was calculated for each classification method. These indicators are based on the confusion matrix with the amount of tress correctly and incorrectly classified per class (Figure 9). In this example, we consider coniferous as positive and deciduous as negative.

	Coniferous (0)	Deciduous (1)
Coniferous (0)	TP	FP
Deciduous (1)	FN	TN

Figure 9. *Representation of the confusion matrix, where TP, TN, FP, FN stands for true positive, true negative, false positive and false negative samples, respectively*

Accuracy indicates de rate of correctly classified samples to all samples. Recall indicates the proportion of true positives to the total amount of samples that should be classified as positive. Precision indicates the ratio of correctly classified positive samples to the total of predicted positive samples. Finally, the F1-score is computed as the harmonic mean of recall and precision measures

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

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

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

$$F1 - score = 2\frac{Precision \ \cdot Recall}{Precision + Recall}$$

3. Results:

nDSM constituted the input data to create segments with individual tree crowns (Figure 10). This elevation raster stores height from ground for the whole project area in an evenly spaced grid (50 cm pixel). Tree crown segmentation was performed over the whole project area, obtaining a vector file with a great number of polygons. From all polygons obtained, a total of 650 coniferous trees were selected among 9 different tree genera.

Due to initially having a much larger number of deciduous trees, only five species were used, adding all of them 954 trees. In total, 1604 polygon trees were used for feature extraction, and ultimately as input in the classification model.

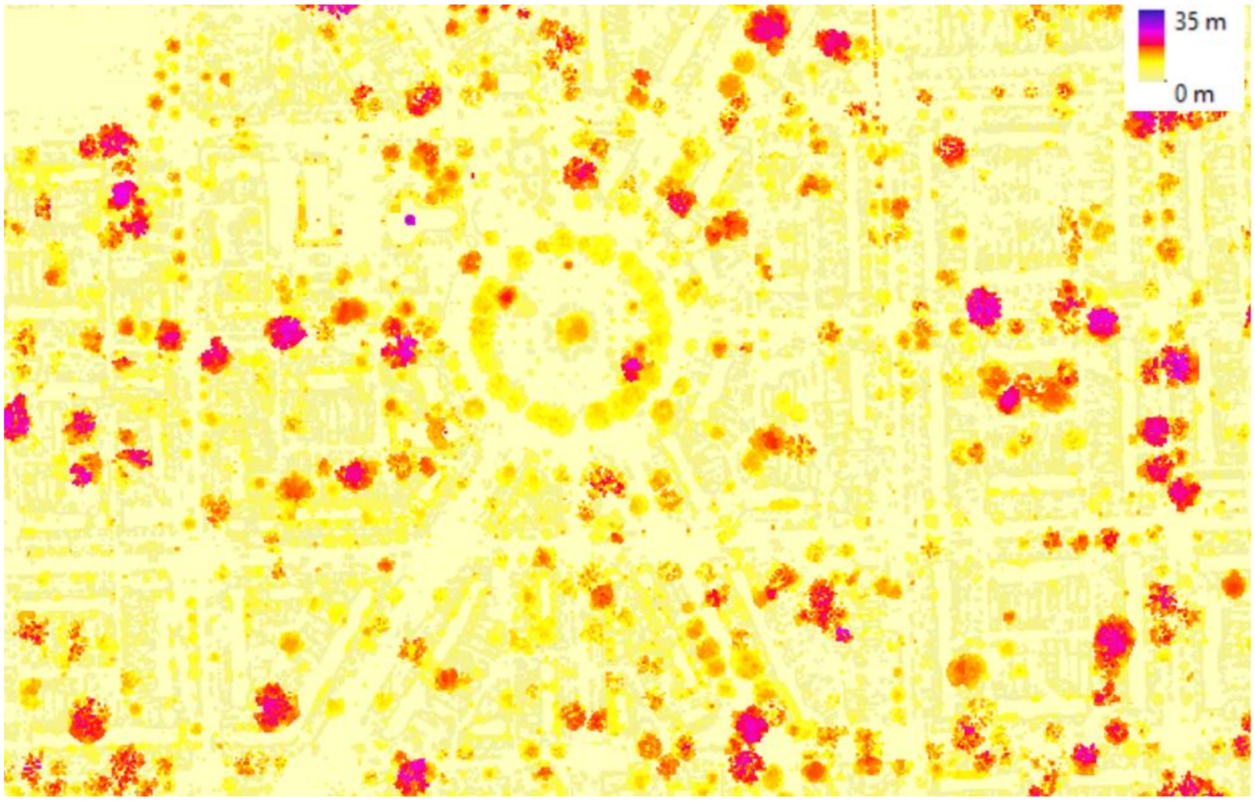

***Figure 10**. Example of the nDSM,raster with color palette representing height above ground*

The segmented individual trees were classified as coniferous or deciduous using the 25 spatial and intensity-related LiDAR features (Table 1). Each classifier tested in this study (MLP, RF, SVM, Logreg and DT) were trained with the same randomly determined training sample set (1122 trees, which represent the 70% of the input data). The remaining samples (482 trees, which was the other 30% of the input dataset) were used to validate the classification performance of the models. According to the obtained overall classification accuracies (Table 2), all proposed methods scored relatively even, between 72 and 76% of samples were correctly classified. DT had a 2-4% lower score than the rest of classifiers.

***Table 2.** Overall accuracy values for the proposed classifiers*

Classifier	Overall accuracy
MLP	74.48%
RF	76.35%
SVM	75.52%
Logreg	74.69%
DT	72.20%

The classification results interpreted by means of recall, precision and F1-score on the test dataset are presented in Figure 11. According to the recall values, all classifiers except for decision tree (DT) scored close or over 0.9 for deciduous, being SVM the highest with 0.935. Nevertheless, coniferous score values were very low for all classifiers, being the lowest also SVM with 0.482. Precision values were more even for most classifiers, except for DT which scored the lowest on coniferous trees with 0.656. Best precision score for coniferous was SVM with 0.829, while best for deciduous was DT with 0.763. Regarding F1-scores, the coniferous were once again worse classified than the deciduous being DT the worst for deciduous (0.773), and MLP the worse for coniferous with 0.614. All the rest of classifiers scored very even for both species.

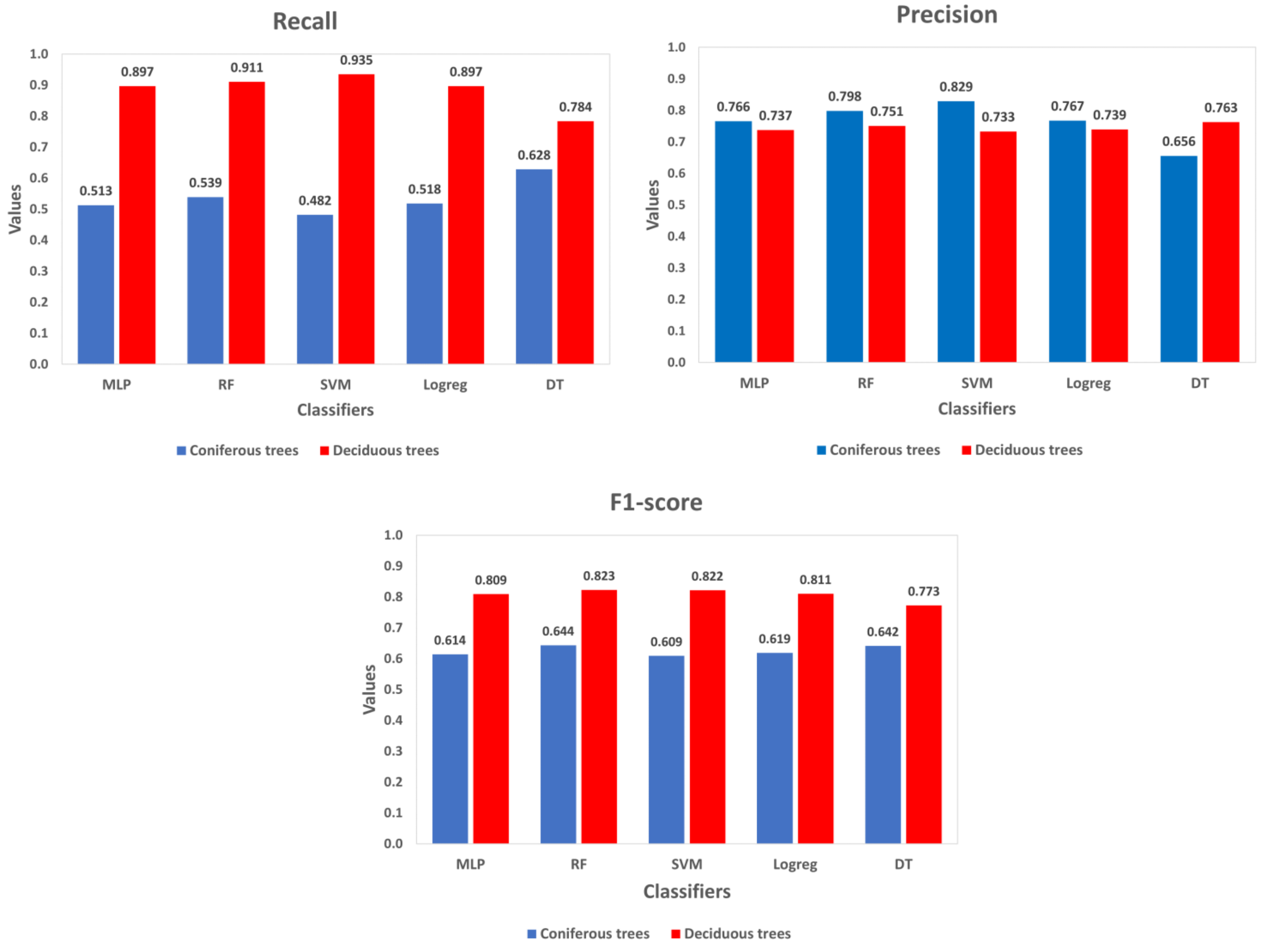

Figure 11. *Recall, precision and F1-score values of MLP, RF, SVM, Logreg and DT classification algorithms for each deciduous and coniferous tree species.*

The classification success of all classifiers was tested with a 10-fold cross-validation method, in order to obtain more robust classification results and avoid overfitting (Table 3). Once again, all classification algorithms got a similar accuracy (73-74%).

Table 3. *10-fold cross-validation results for all classification methods*

Classifier	Overall accuracy
MLP	73.89%
RF	73.51%
SVM	71.14%
Logreg	73.07%
DT	74.32%

A feature importance analysis was performed to see which ones of the 25 spatial and intensity-related features have a bigger impact on the RF classification (Figure 12). The importances are calculated as the mean accumulation of the impurity decrease with each tree. Intensity-based features are clearly more important features in the classification in comparison with spatial-based. The lowest percentile of intensity (5th) scored the highest, followed by the 25th, 50th, 75th percentiles, mean intensity, and 90th percentile of intensity. Among the spatial-based features, standard deviation of Z scored the highest, and the rest of features have evenly low importance.

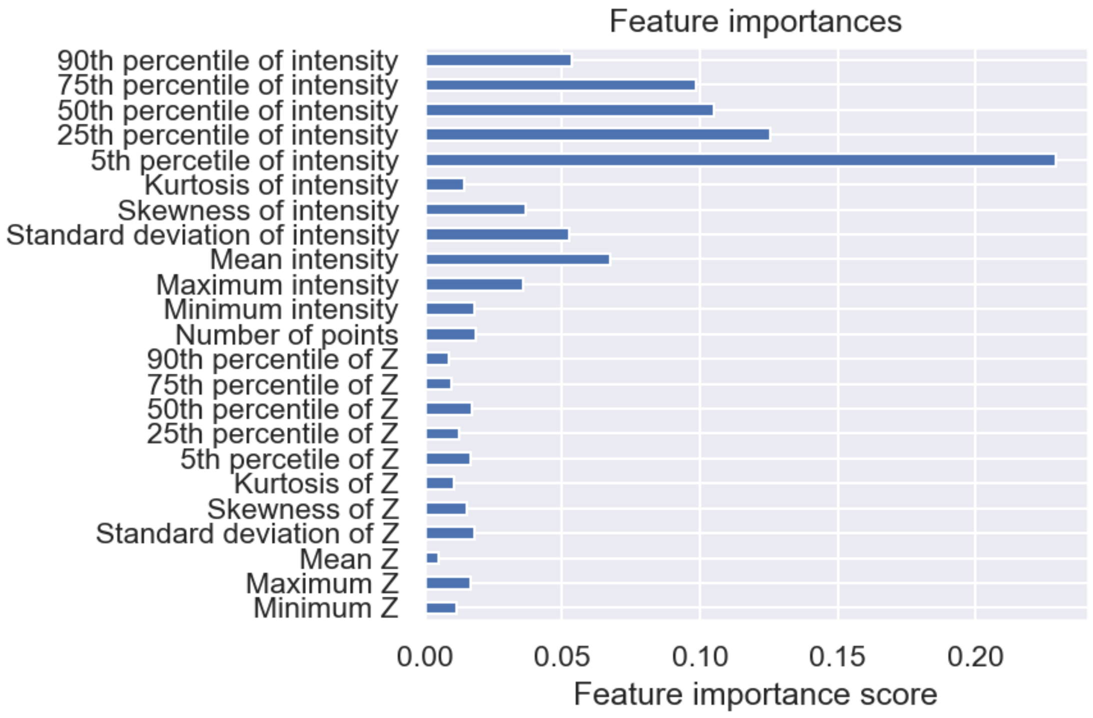

Figure 12. *Feature importance scores in percentage in the RF classification*

4. Discussion and conclusions

Five widely used machine learning algorithms (MLP, RF, SVM, logreg and DT) were used to assess tree species classification (coniferous/deciduous) in an urban environment, using raw 3D LiDAR data. The results show similar performances for all the classifiers, although DT provided a slightly lower accuracies (Tables 2 and 3). Nevertheless, the classified success in deciduous are greater than on coniferous (Figure 11).

Cetin and Yastikli (2022) computed tree crown segmentation by directly classifying the point cloud using a mean shift clustering algorithm. The present study used the whole LiDAR dataset in the District of Columbia, a much larger project area than in Cetin and Yastikli (2022). Therefore, an ALS-derived Canopy Height Model (CHM) was used instead of the mean shift clustering algorithm, to perform segmentation. This is a much less time-consuming method, used as well in other similar studies (Ørka et al. 2009). The maximum overall classification accuracy in Cetin and Yastikli was 84% using RF, while Ørka et al. (2009) reached an 88%, although they used a linear discriminant analysis (LDA) method. In addition, Ørka et al. (2009) used full-waveform LiDAR features, not available in the present study. The maximum overall accuracy in the present study was 74.4%, achieved by the random forest classifier.

The number of train and test samples in the study area was higher than in Cetin and Yastikli (2022), although still relatively small compared with Ørka et al. (2009). There was available a much larger number of deciduous samples, but it was used the totality of available coniferous samples after manual validation (see 2.5.3). It was decided to randomly select a lower number of coniferous to somehow much the number of samples of coniferous, and therefore obtain meaningful overall accuracy scores.

The results of feature importance are noticeably different than in Cetin and Yastikli (2022). In this previous study, the spatial-based features had more importance in their models. Just the opposite happened in the present work, more based on intensity features (Figure 12). The reason for this can be the quality in the input data itself. In Cetin and Yastikli (2022) they acquired the ground-truth data in a small area through field-investigation and photo interpretation. They were able to validate all samples so there would not be errors in the dataset. However, in the present study it was used an open-source dataset, where it wasn't possible to validate the position of the trees, the quality of their tree species classification, etc. Because of the tree composition of the city of Washington DC (few coniferous in comparison with deciduous, and spread over the whole area), it was necessary to use the whole city as the project area, to have enough coniferous samples. Because of this reason, a different method for tree crown segmentation was implemented (see 2.5.4). There are also differences in the LiDAR datasets used in Cetin and Yastikli (2022) and the present study. The point density of 16 points/m^2 offer more information on the internal structure of trees than the 4.5 points/m^2 in this study. The energy reflected by objects could also be affected by different LiDAR sensor and flying height, making the comparison between both studies more difficult.

Regarding classification methods, there is other algorithms not tested in the present study that could improve the results, although the writer doesn't think it will be a dramatical increase in accuracy. From other ensemble methods like extreme boosting (XGBoost), to simpler K-nearest neighbor (KNN) algorithm. Although, the efforts should go more on the direction of having better, validated input data.

This study shows that it is possible to use open-source data that could be found in several places around the world as input for classification of urban tree species. LiDAR data is becoming available as open data in more countries. Open-source urban trees datasets, with position and tree species information exist as well in several locations through crowd source. However, it should be possible to perform this study using just open-source tools, without the need of commercial software. In the present study, commercial software was used for pre-processing of ALS data and tree crown segmentation. Some other methods, perhaps integrated in the same Python environment should be inspected. For example, PDAL (PDAL Contributions, 2020) is an open-source library for translating and manipulating point cloud data, that could be used for pre-processing of the ALS dataset. PyCrown (Zörner et.al., 2018) is a raster-based individual tree segmentation for LiDAR data, that could be implemented for the tree segmentation in this study.

Several studies have stated that fusing LiDAR with optical imagery could achieve better tree species classification accuracy than using either dataset individually (Wang and Liu, 2018). Full wave-form LiDAR features could improve the classification accuracy as well (Ørka et al. 2009), although this feature is often not available in LiDAR datasets.

Considering the uncertainty of the input data, the classification results from this study showed a moderate success, especially on classifying deciduous trees. Compared with traditional field surveys and aerial photograph interpretation, the proposed machine learning approach has obvious benefits, in terms of cost and speed of classification.

5. Appendix

5.1. Source code and test results – GitHub

The python source code and test results are gathered in a GitHub repository:

https://github.com/luisbart/Projects_STK4155/tree/main/Project3

01_hyperparameter_tuning.py: source code containing the test for tuning hyperparameters in all classifiers

02_UrbanTrees_clf.py: classification for obtaining overall accuracy, 10 fold cross-validation accuracy, several figures (confusion matrix, ROC curve, cumulative gain curve, plot of a decision tree, feature importance in RF), F1/precision/recall scores and a fast test with many classifiers (LazyClassifier).

5.2. Supplementary figures

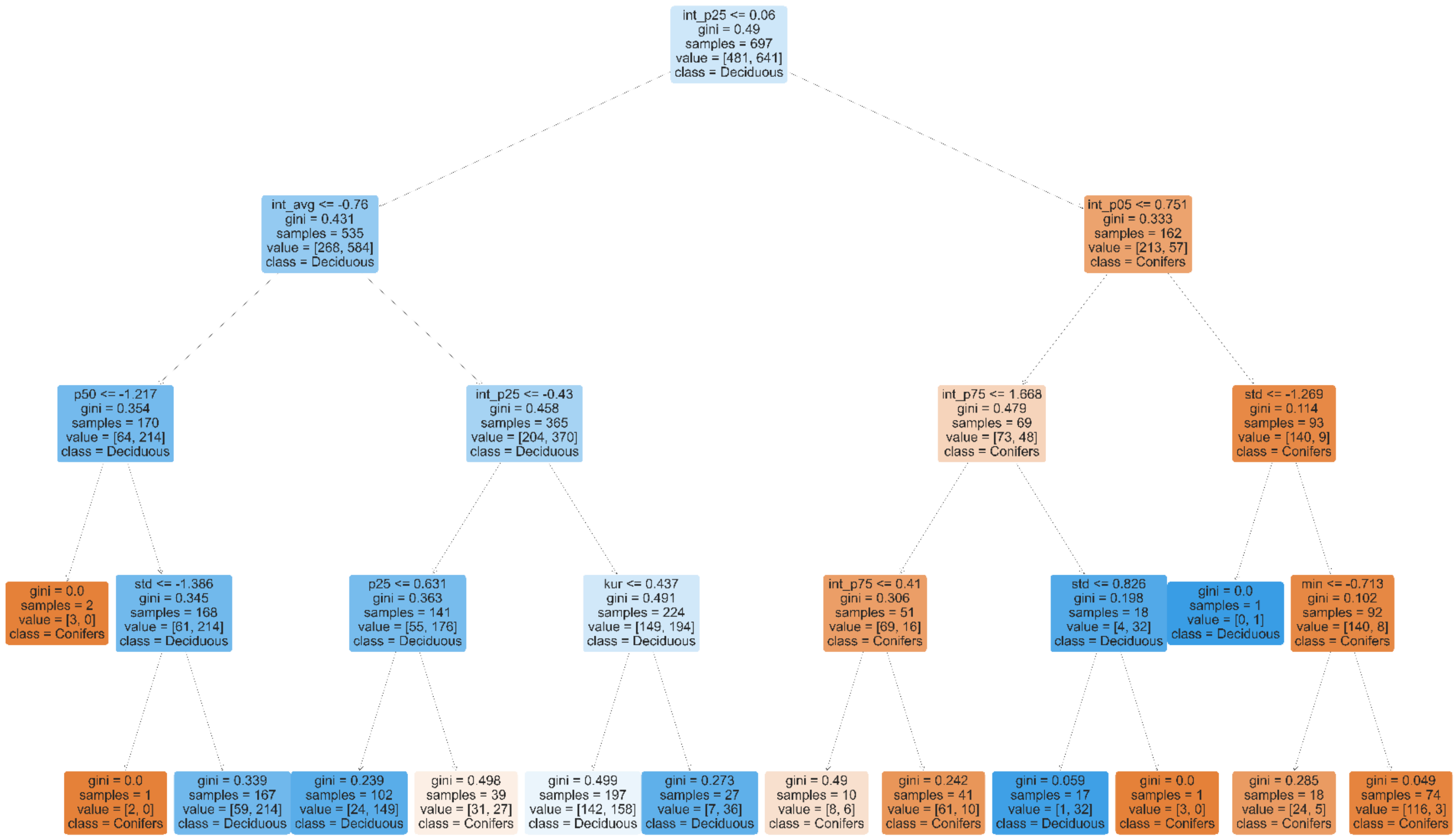

Figure A1. *Diagram of a DT used for urban tree classification*

6. References

ASPRS (The American Society for Photogrammetry & Remote Sensing (2013). LAS Specification Version 1.4 – R13. https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities (Accessed 10 December 2022).

Cetin, Zehra & Yastikli, Naci. (2022). The Use of Machine Learning Algorithms in Urban Tree Species Classification. *ISPRS International Journal of Geo-Information. 11. 226. 10.3390/ijgi11040226.*

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

Hastie, T., Tibshirani, R. and Friedman, J. (2009) The Elements of Statistical Learning: Data mining, Inference, and Prediction. *2nd Edition, Springer Series in Statistics, Springer, New York.*

Hjort-Jensen, M. (2022). Applied Data Analysis and Machine Learning. https://compphysics.github.io/MachineLearning/doc/LectureNotes/_build/html/intro.html. (Accessed 14 December 2022)

Isenburg, M., 2022. LAStools. Retrieved from https://lastools.github.io/ (Accessed 10 December 2022).

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. 56-61. *10.25080/Majora-92bf1922-00a*.

Oke, T. R., 1982: The energetic basis of the urban heat island. Quart. J. Roy. Meteor. Soc., 108, 1–24.

PDAL Contributors, 2020. PDAL Point Data Abstraction Library. doi:10.5281/zenodo.2556737

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., and Thirian, B. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning, v. 12, p. 2825-2830.

QGIS Development Team (2022). QGIS Geographic Information System. *Open Source Geospatial Foundation Project*. http://qgis.osgeo.org (Accessed 10 December 2022).

Rouault, E., Warmerdam, F., Schwehr, K., Kiselev, A., Butler, H., Łoskot, M., Szekeres, T., Tourigny, E., Landa, M., Miara, I., Elliston, B., Kumar, C., Plesea, L., Morissette, D., Jolma, A., Dawson, N., (2022). GDAL. https://zenodo.org/record/6801315#.Y5YqQH2ZPq5 (Accessed 11 December 2022)

Trimble Germany GmbH (2021). Trimble Documentation eCognition Developer 10.2 Reference Book; *Trimble Germany GmbH: Munich, Germany*.

United Nations, Department of Economic and Social Affairs, Population Division (2019). World Urbanization Prospects: The 2018 Revision (ST/ESA/SER.A/420). *New York: United Nations.*

Wang, Kepu & Liu, Xuehua. (2018). A Review: Individual Tree Species Classification Using Integrated Airborne LiDAR and Optical Imagery with a Focus on the Urban Environment. Forests. 10. 10.3390/f10010001.

Waskom, Michael. (2021). Seaborn: statistical data visualization. Journal of Open Source Software. 6. 3021. https://doi.org/10.5281/zenodo.883859 (Accessed 10 December 2022).

Zörner, J.; Dymond, J.; Shepherd J.; Jolly, B. PyCrown - Fast raster-based individual tree segmentation for LiDAR data. Landcare Research NZ Ltd. 2018, https://doi.org/10.7931/M0SR-DN55

Ørka, H. & Næsset, E. & Bollandsås, O. (2009). Classifying species of individual trees by intensity and structure features derived from airborne laser scanner data. Remote Sensing of Environment. 113. 1163-1174. 10.1016/j.rse.2009.02.002.