University of Oslo

**Project 3**

**FYS-STK4155**

José Luis Barreiro Tomé

December, 2022

Abstract:

1. Introduction
2. Data and methods
   1. Theory
      1. Neural Network and Logistic regression

The reader is referred to section 1.1.2 and 2.4 for in [Project 2](https://github.com/rcorseri/UiO/tree/main/Project2/Report) report (<https://github.com/rcorseri/UiO/tree/main/Project2/Report>), for Neural network and logistic regression, respectively.

* + 1. Decision trees

Decision trees 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 with contains the most information regarding the target feature.

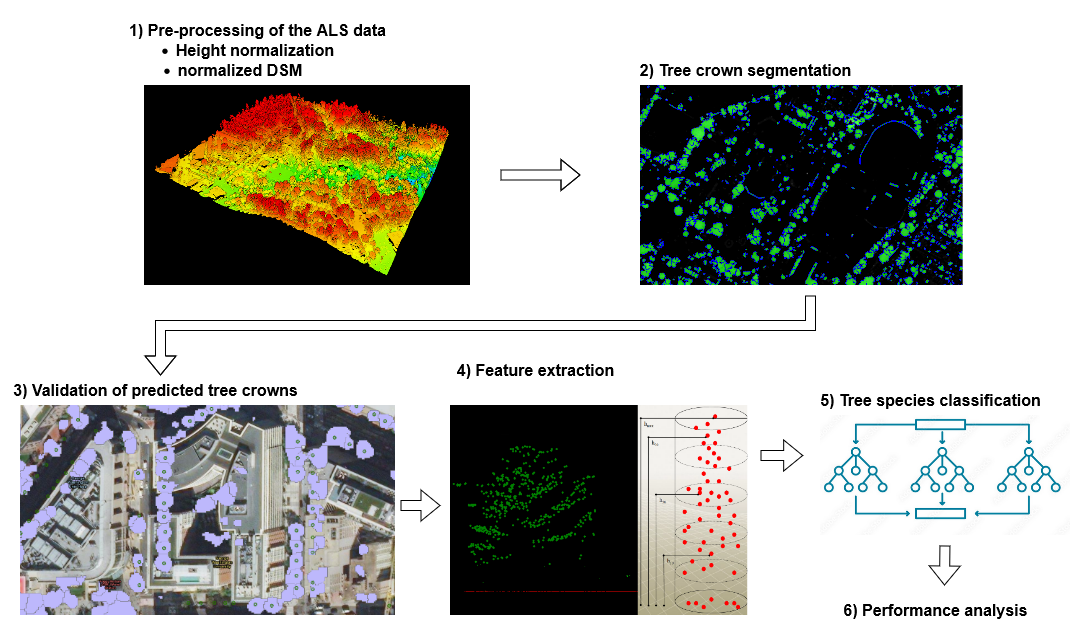
* + 1. Random Forest
    2. Suport Vector Machines
  1. Study area and data

The LiDAR point-cloud data is also 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 conducting to set all points in different classes, such as Ground, low vegetation, medium vegetation, high vegetation, buildings, water etc.

Ground-truth data for tree species come from the Washington DC Urban Forestry Street Trees, 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.

* 1. General workflow

The main steps taken in this work for tree species classification of urban trees using 3D LiDAR data are visualized in Figure 1 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 in ALS data preprocessing. 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) ans seaborn (Waskom, 2020).



***Figure 1.*** *Flowchart representing general workflow of this study*

* + 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, an elevation grid was produced in raster format, with 50x50 centimeters raster cell (or pixel). This raster constituted the normalized Digital Surface Model (nDSM).

* + 1. 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.

* + 1. 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, 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.

* + 1. 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 |
| Skewnes os 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 |

* + 1. Tree species classification

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

MLP: number of nodes, number of layers, activation function, solver, learning rate, lambda:

another way of doing this might be:

- plot MSE vs number of nodes for a NN with one hidden layer for a couple of chosen learning rates

- plot MSE vs depth for NN with a fixed number of hidden nodes per layer for a couple of chosen learning rates

This would convey the pretended message in a more concise and easier to read way.

Choose 1 layer, 1 neuron, eta=1e-5, lmbda=10

Logistic: default penalty: l2. Default solver: lbfgs. Keep both

Decision tree. Default criterion: gini. Test entropy and log\_loss. Default splitter: best. Default max\_depth: none. Keep all

Entropy works a bit better 0.64 vs 0.66

SVM. Default C=1 (test several). Default kernel: rbf. Default gamma: ‘auto’ (test scale)

C=1 works best. Gamma same score with both

RF. Default criterion: gini. Test entropy and log\_loss. Default splitter: best. Default max\_depth: none. Keep all

Gini and entropy same score

* + 1. Performance analysis

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 2). In this case, we consider coniferous as positive and deciduous as negative.

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

***Figure 2.*** *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

1. Results and discussion

nDSM constituted the input data to create segments with individual tree crowns (Figure 3). This elevation raster stores height from ground for the whole project area in an evenly spaced grid (50 cm pixel). From all polygons obtained during segmentation, 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 as input in the classification models.

Background pattern

Description automatically generated with medium confidence

***Figure 3****. Screenshot of the nDSM, with color palette representing height above ground*

Each classifier tested in this study (MLP, RF, SVM, Logistic Regression and decision tree) 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 30% of the input dataset) were used to validate the classification performance of the models.

1. Conclusion
2. 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.*

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*.

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*.

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).