Course Title: Environmental Data Analytics.
Degree Program: Master's in data science.
Instructor: Mohammad Mahdi Rajabi

Final Project: LiDAR-Based Tree Detection in Mixed Forests

Deadline: 20 January 2025

Dataset Description

The dataset is sourced from a UAV LiDAR survey and includes a dense point cloud (37 points/m²) covering a mixed and dense forest in Perm Krai, Russia. It also includes a field inventory with 3600 trees recorded across 10 rectangular ground plots (100 m x 50 m). The dataset provides:

- **Point Cloud Data:** Preprocessed LiDAR point clouds with height normalization (Z-coordinate represents height above ground).
- **Field Inventory Data:**
    - Tree species labels for all recorded trees.
    - Diameter at breast height (DBH) for all trees.
    - Tree height measurements for ~20% of trees.
    - Tree age data for ~10% of trees.

The dataset is freely available on Kaggle (https://www.kaggle.com/datasets/sentinel3734/tree-detection-lidar-rgb/data), accompanied by example Python notebooks.

To ensure diversity and uniqueness among group projects, **each group must select one item from Table 1** to focus on as part of their project. Each item can only be chosen by one group, so you are encouraged to negotiate amongst yourselves to finalize the selections. Once all groups have agreed, one designated representative from the class should submit the complete list of items chosen by each group. Additionally, your final project report must include a **dedicated section** that clearly outlines the logic and detailed implementation of your selected item, highlighting its integration into the overall project pipeline and its contribution to achieving the project goals.

Detecting individual trees from the LiDAR point cloud data

The primary task is to develop a pipeline for **detecting individual trees** from the LiDAR point cloud data. The goal is to identify the locations of tree tops within the forest plots and compare the detected trees with the ground truth field inventory data.

The detection pipeline should:

1. **Preprocess the Data:**
    - Load the LiDAR point cloud data and normalize it if needed.
    - Filter irrelevant points (e.g., ground points below a height threshold) and remove residual noise.
    - Partition the dataset into training and testing sets.

2. **Implement and Extend Detection Algorithms:**
    - Start with the provided **local maxima filtering algorithm** for baseline tree detection.
    - Then design and implement an **advanced DNN architecture** suitable for point cloud data.
        - Suggested architectures include PointNet, PointNet++, or DGCNN (**unless the item you choose from Table 1 includes a specific DNN model architecture**).
    - Train the model to detect individual tree locations using the field inventory as ground truth.
3. **Evaluate the Results:**
    - Match detected trees with ground truth data using the provided matching algorithm.
    - Calculate and report key performance metrics:
        - Precision: Proportion of correctly detected trees among all detected.
        - Recall: Proportion of ground truth trees correctly detected.
        - F1-Score: Harmonic mean of precision and recall.
        - Distance Error: Average distance between detected tree tops and ground truth tree locations.
4. **Comparison with Baseline:**
    - Compare the DNN's performance to the baseline local maxima filtering algorithm.
    - Highlight improvements in detection accuracy and analyze limitations.

Expected Outcomes

1. **Code:**
    - A script or notebook implementing the preprocessing pipeline and DNN model.
    - Clear documentation.
2. **Evaluation Results:**
    - Detailed performance metrics (e.g., precision, recall, F1-score, and distance error) for the DNN model and baseline comparison.
    - Analysis of model behavior in different forest plots.
3. **Visualizations:**
    - Plots or 3D visualizations showing detected tree locations overlaid on the point cloud.
    - Comparative visualizations of results from the DNN model and the baseline.
4. **Report:**
    - A structured report summarizing:
        - Data preprocessing steps.

- Model design and training process.
- Performance evaluation and insights.
- Limitations and potential improvements.

Table 1. List of Tasks to Choose From.

Item No.	**Focus Area**	**Description**
1	Convert Point Cloud to Graph (KNN)	Use **K-Nearest Neighbors (KNN)** to construct a graph from the point cloud and apply a Graph Neural Network (GNN), specifically the **Point-GNN** for tree detection.
2	Convert Point Cloud to Graph (Radius-Based Neighbors)	Use **radius-based neighbors** to construct a graph from the point cloud and apply a GNN, specifically **DGCNN** (with EdgeConv) to detect trees.
3	Convert Point Cloud to 2D Slices	Convert the point cloud into 2D slices (e.g., horizontal or vertical cross-sectional images) and apply a deep learning model such as **ResNet** or **UNet** to identify individual trees.
4	Convert Point Cloud to Depth Image	Convert the point cloud to a 2D depth image and apply a deep learning model such as **ResNet** or **UNet** to identify individual trees.
5	Convert Point Cloud to Voxel Grid	Voxelize the point cloud to a 3D grid and apply a deep neural network such as **3D CNN** to process volumetric data and detect individual trees.
6	Preprocess: Remove Outliers using Clustering-Based Removal (Non-ML)	Use a clustering algorithm like **DBSCAN** to identify and remove outliers that do not belong to sufficiently large clusters, cleaning the dataset prior to tree detection.
7	Preprocess: Downsample	Perform downsampling using **Farthest Point Sampling (FPS)** and **Voxel Grid Filtering** to reduce data density for efficient processing before tree detection. Compare the impact of these two

Item No.	Focus Area	Description
		downsampling methods on the quality of tree detection results
8	Attention-Based Networks: Point Transformer	Use the **Point Transformer** architecture, an attention-based model designed for raw point cloud data, to detect trees.
9	Attention-Based Networks: PointASNL	Use the **PointASNL (Point Attention Sampling and Neighborhood Learning)** architecture for detecting trees directly from raw point cloud data.
10	RandLA-Net	Use the **RandLA-Net** architecture for detecting trees directly from raw point cloud data.
11	Synthetic Point Cloud Generation for Augmentation **(this task can be done by a group of two or three)**	Use **PointFlow** to generate synthetic point clouds that resemble the real dataset, introducing variability in tree structures and arrangements. Combine the real and synthetic point clouds to train a tree detection model, enhancing its robustness and generalization.