Presentation and Final Report items to include:

Name(s) including team members (if any): Shaman Garcia

Title: Biomass analysis using family level tree identification

Geography/Area of Interest: Saginaw Forest

Research Question: How can we use aerial imagery and forest classification in order get a biomass estimate for Saginaw Forest?

Field Data (what was used and collected? did you add to existing field data or collect all new field data? describe your field campaign and methods.):

- Saginaw community composition layer
- Ground points from EAS 447
- Aerial imagery from lab 3

Other Data including Remote Sensing Data:

- Lab 3 data mentioned above

Description of context and methods:

- Pycrown canopy identification - https://github.com/manaakiwhenua/pycrown
    - Run pycrown on a CHM, DSM, LAS in order to get a 3D model of canopy
    - Get the tree centers and indices
- Image classification of the NAIP imagery
    - Using the pixel percentages, create ratios of broadleaf, evergreen, and mixed forest
    - Multiply these ratios by the canopy cover model to get the biomass of the canopy
- Allometric equations
    - Use number of trees from pycrown and allometric equations for trunk biomass
    - Use EAS 447 data to get non-tree biomass

Analysis and Results and Discussion:

- Biomass estimate from pycrown
- Pixel ratios from classification
- Allometric equations used

Conclusions including what you would do differently:

Segment imagery into more manageable sections

Detailed Methods:

Pycrown - AI Generated

AI generation with UMICH GPT was used to create a detailed list for the steps that I used. When I would encounter errors with the code when using the terminal, I would ask UMICH GPT about the errors and it would give me the next step on how I can resolve the problem. This list is modified to be able to replicate the code without the errors.

Step 1: Install Anaconda (If Not Already Installed)

1. Download Anaconda from the Anaconda Individual Edition page.
2. Install it using default settings.
3. Ensure that you can access the Anaconda Prompt.

Step 2: Set Up the Conda Environment

1. Open the Anaconda Prompt.

Remove any pre-existing environment named `pycrown-env` to avoid conflicts:
bash
Copy code

```
conda deactivate
conda remove -n pycrown-env --all
```

2.

Create a new environment with Python 3.6:
bash
Copy code

```
conda create -n pycrown-env python=3.6
```

3.

Activate the new environment:
bash
Copy code

```
conda activate pycrown-env
```

4.

Step 3: Configure Conda for Compatibility

Add the `conda-forge` channel to prioritize compatible packages:
bash
Copy code

```
conda config --add channels conda-forge
conda config --set channel_priority strict
```

1.

Install the required packages:
bash
Copy code

```
conda install gdal geopandas rasterio laspy numpy scipy pandas shapely
scikit-image scikit-learn cython numba
```

2. This ensures you have compatible versions of all dependencies.

Step 4: Address Specific Issues

Fix `dataclasses` Missing Module Issue

Since Python 3.6 does not include `dataclasses` by default, install the backport:

bash
Copy code

```
pip install dataclasses
```

Fix `SimpleQueue` ImportError

`SimpleQueue` is unavailable in Python 3.6, so install a compatible version of `laspy`:

bash
Copy code

```
pip install laspy==1.5.1
```

Step 5: Download and Set Up PyCrown

Clone the PyCrown repository or download it as a ZIP file:
bash
Copy code
```
git clone
https://bitbucket.org/zoernerj-landcare/tree-segmentation.git
```

1.

Navigate to the PyCrown directory:
bash
Copy code
```
cd tree-segmentation/pycrown
```

2.

Install PyCrown using `setup.py`:
bash
Copy code
```
python setup.py install
```

3.

Step 6: Test Installation

Verify that PyCrown is installed correctly:
bash
Copy code
```
python setup.py test
```

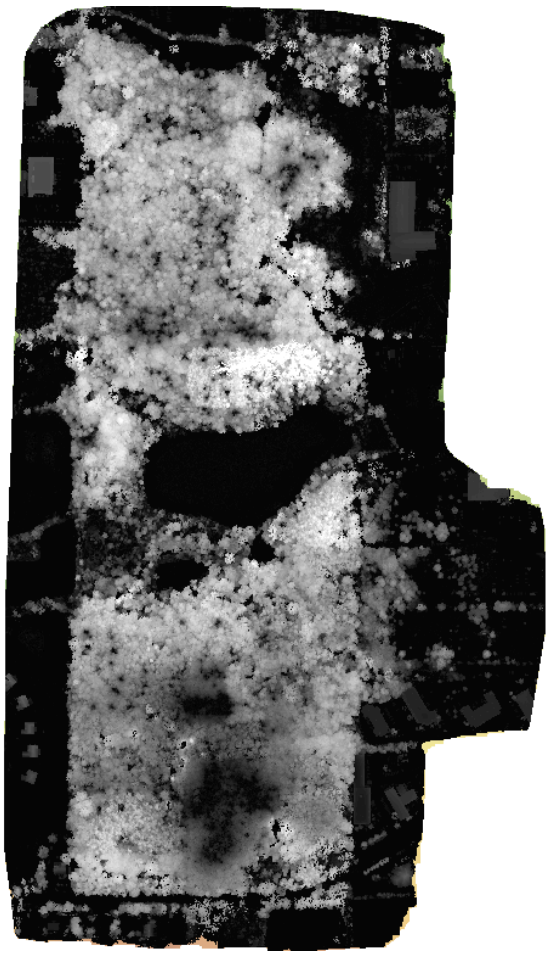
1. Resolve any errors if encountered.

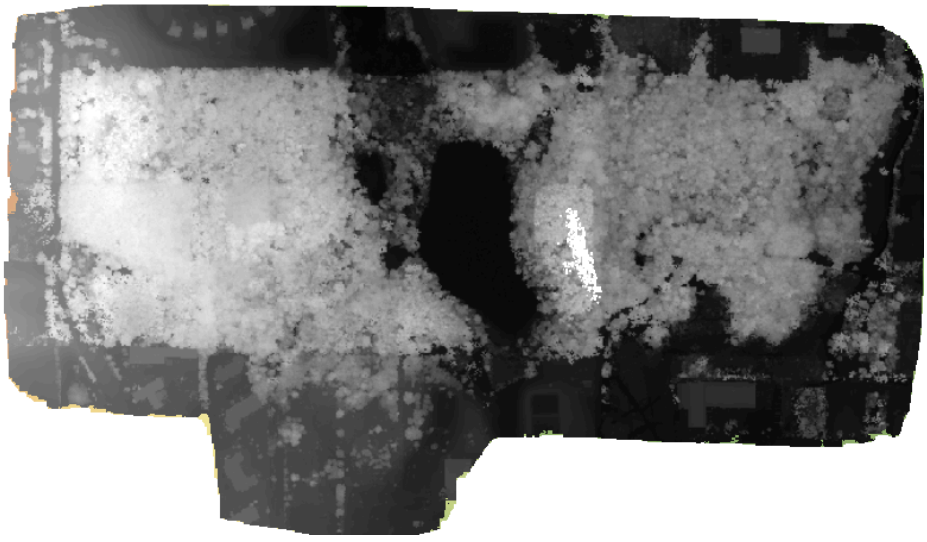
Check that all dependencies are working by testing imports:
bash
Copy code
```
python
```
Inside the Python interpreter, run:
python
Copy code
```
import geopandas
import laspy
import fiona
print("All dependencies imported successfully!")
```

2. If no errors occur, exit the interpreter with `exit()`.

Step 7: Run the Example Script

Here, you would replace example with the file directory containing the three input files(DSM, DTM, CHM)





DSM - DTM = CHM

Navigate to the `example` directory:
bash
Copy code
```
cd ../example
```

1.

Run the example script:
bash
Copy code
```
python example.py
```

2.
    - This will process the provided sample data.
    - Check the `result` folder for output files (`tree_tops.shp`, `tree_crowns.shp`, etc.).
3. If errors occur, review the error messages and ensure the following:
    - Your environment includes all required dependencies.
    - The `laszip` tool is installed if working with `.laz` files.

Step 8: Process Your Own Data

1. Place your CHM, DSM, and DTM files in a directory, e.g., `my_data`.
2. Copy the `example.py` script and rename it, e.g., `my_analysis.py`.

Modify `my_analysis.py` to point to your data files:
python
Copy code
```
chm_path = 'my_data/your_chm.tif'
dsm_path = 'my_data/your_dsm.tif'
dtm_path = 'my_data/your_dtm.tif'
```

3.

Run the modified script:
bash

Copy code
```
python my_analysis.py
```

4.

Step 9: View Outputs in ArcGIS Pro

1. Open ArcGIS Pro and load the output shapefiles:
    - `tree_tops.shp`
    - `tree_crowns.shp`
2. Visualize and analyze the results.

Common Troubleshooting

1. **DLL Load Errors (e.g., GDAL or Fiona):**
    - Ensure GDAL and Fiona are installed from `conda-forge`.

Check GDAL functionality:
bash
Copy code
```
gdalinfo --version
```

    -
2. **Python Version Issues:**

Confirm the Python version:
bash
Copy code
```
python --version
```

    -
    - It must be 3.6 to match PyCrown requirements.
3. **Missing Tools for `.laz` Files:**

Install `laszip` from `conda-forge`:
bash
Copy code
```
conda install -c conda-forge laszip
```

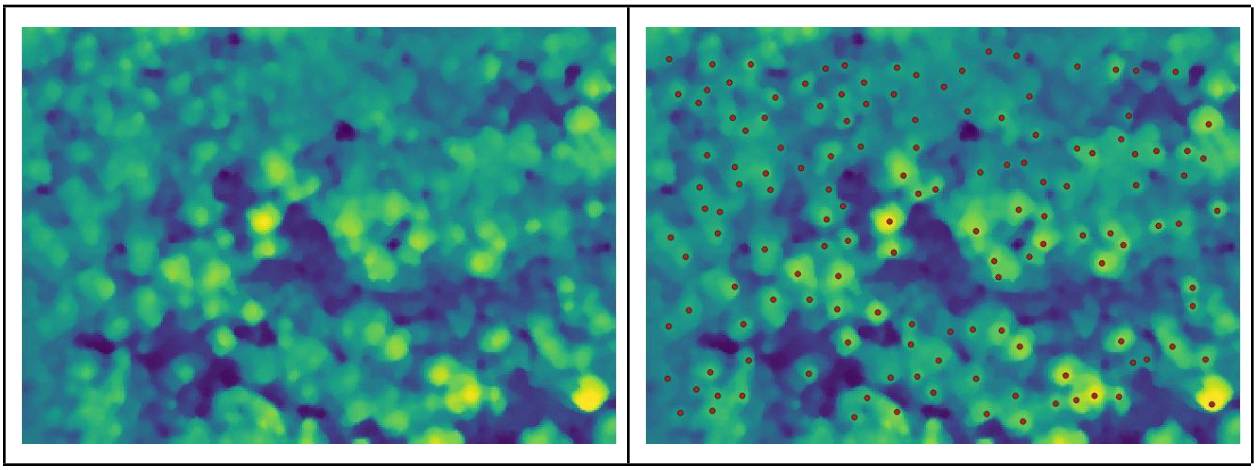
    -

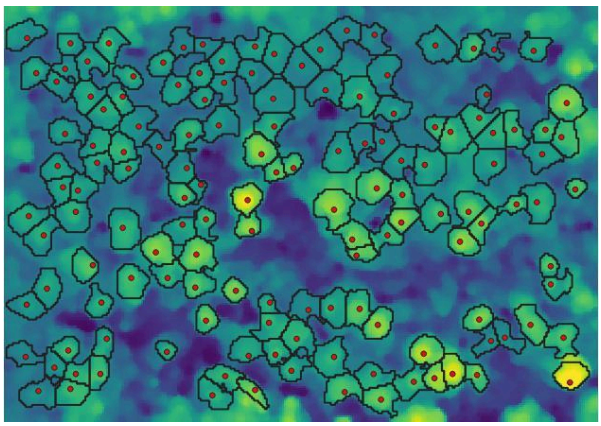
Final Notes

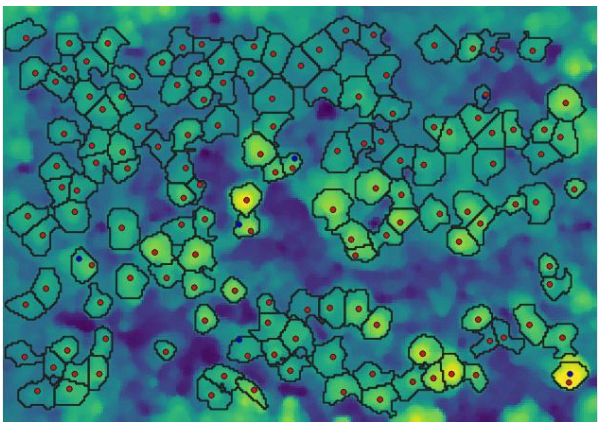
- Always create a clean Conda environment to avoid dependency conflicts.
- Stick to the `conda-forge` channel for consistent package versions.
- Update `requirements.txt` or `setup.py` as needed if you encounter new dependency issues.

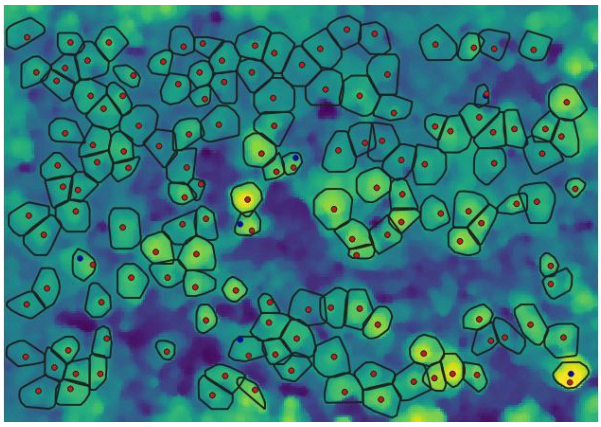
Output - Example

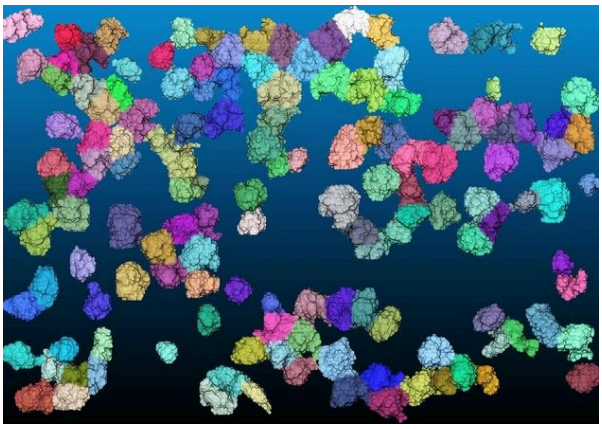
```
(pycrown-env) C:\Users\shama\Desktop\516\personal project\pycrown\example>python example.py
Tree crowns delineation: 0.001s
Number of trees: 170
Tree tops corrected: 13
Tree tops corrected: 7.6470588235294 12%
DSM correction: 8
COM correction: 5
Converting LAS point cloud to shapely points
Converting raster crowns to shapely polygons
Attach LiDAR points to corresponding crowns
Create convex hull around first return points
Classifying point cloud
Number of trees detected: 149
Processing time: 0:00:20.123166 [HH:MM:SS]
```

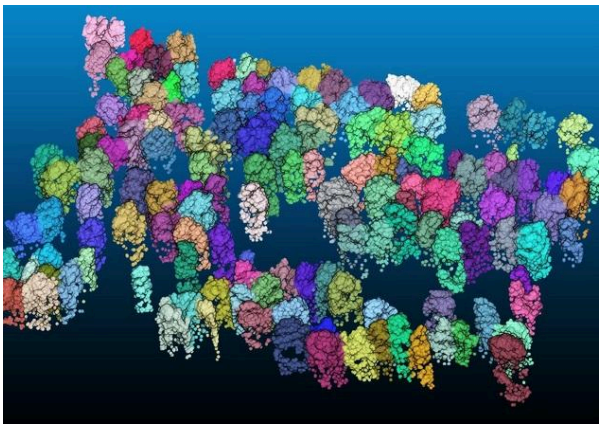


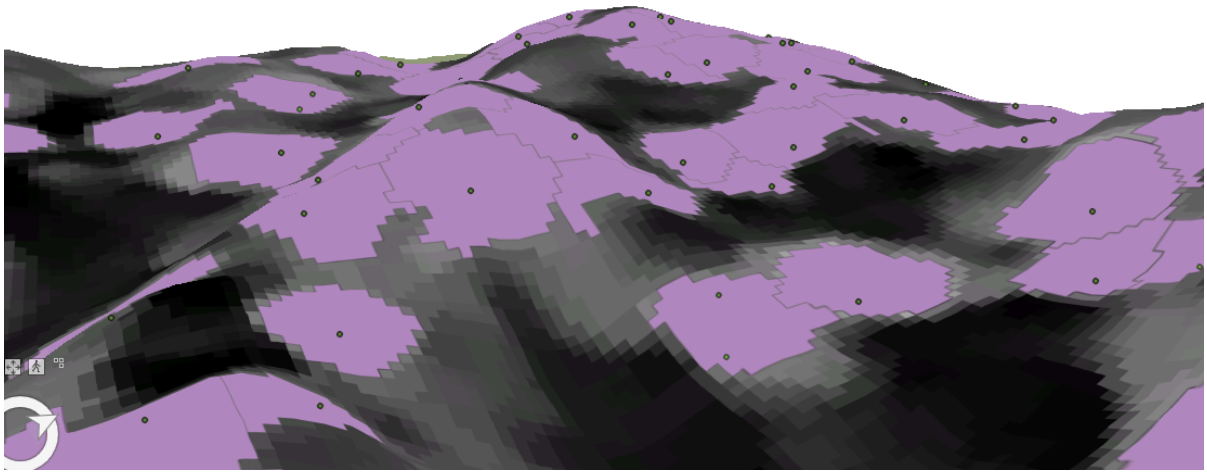












AI Generated using UMICH GPT in order to create a fleshed out list that was generated based on a list of steps and details I input into the chatbot.

Step-by-Step Guide for NAIP Image Segmentation and Classification

This guide assumes:

- You are using **ArcGIS Pro** with the **Image Analyst** extension.
- You have a NAIP image as your input raster.
- You have training data for landcover community types.

Step 1: Segment the NAIP Image

1. Open your NAIP image in **ArcGIS Pro**.
2. Navigate to the **Image Analyst** toolbar.
3. Select **Segment Mean Shift** from the **Raster Functions** menu.
4. Configure the parameters:
    - **Spectral Detail**: 15
    - **Spatial Detail**: 15
    - **Minimum Segment Size in Pixels**: Leave as default (or adjust based on your project's needs).
5. Click **Run** to execute the segmentation.
6. Verify that the segmented image layer appears in the **Contents Pane**.

Troubleshooting:

- If segmentation results are overly generalized or fragmented:
    - Decrease or increase **Spectral Detail** or **Spatial Detail** based on your input raster's spectral variability.
    - Ensure the input image has sufficient resolution and no data gaps.

Step 2: Prepare the Classification Wizard

1. In the **Image Analyst** tab, open the **Classification Wizard**.
2. Select the segmented NAIP image as your input raster.
3. Choose the **Supervised Classification** method.
4. Ensure your training samples are ready:
    - Use **Landcover Community Type** for Saginaw Forest data to train the classification.
    - If necessary, digitize polygons or points representing each class and save them as a feature class.
5. Set your training samples feature class under **Training Data**.

Troubleshooting:

- If training data does not load:
    - Confirm that the training samples feature class is in the same projection as the input raster.
    - Ensure that each sample is labeled with the appropriate landcover class.

Step 3: Run the Classification Wizard

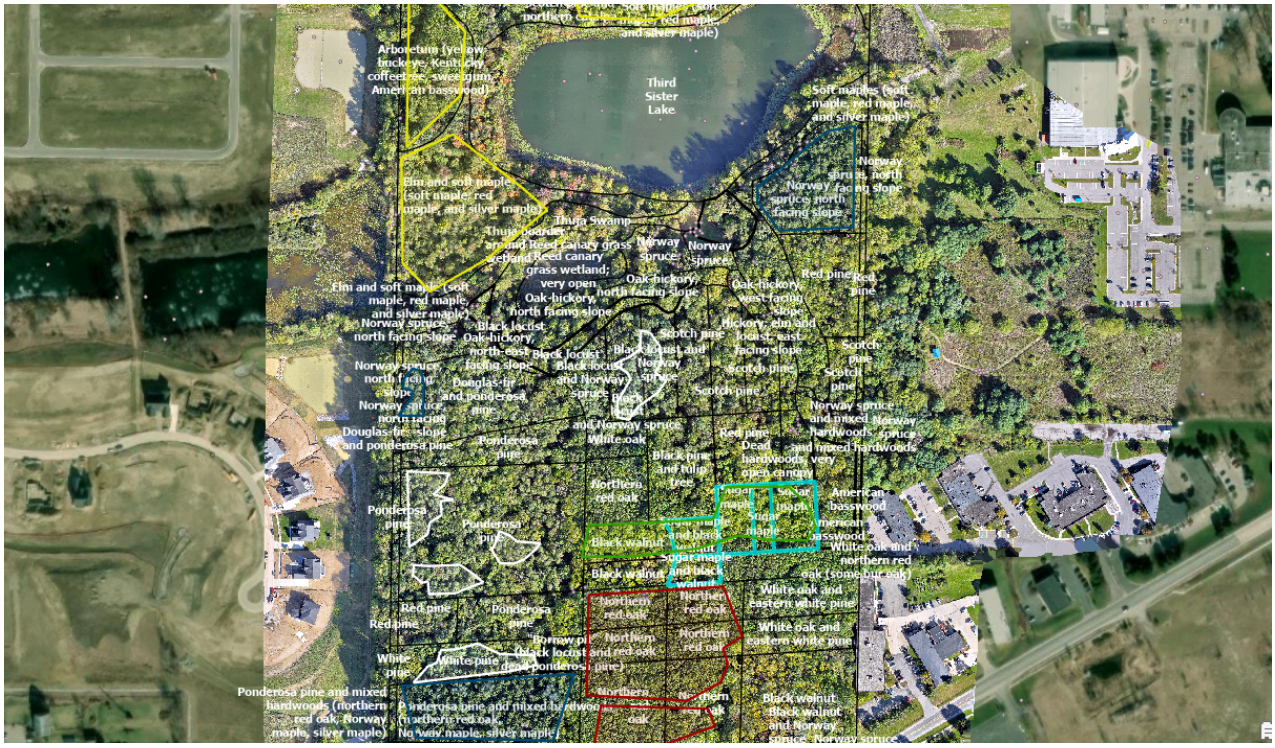
In the **Classification Wizard**, verify all settings:

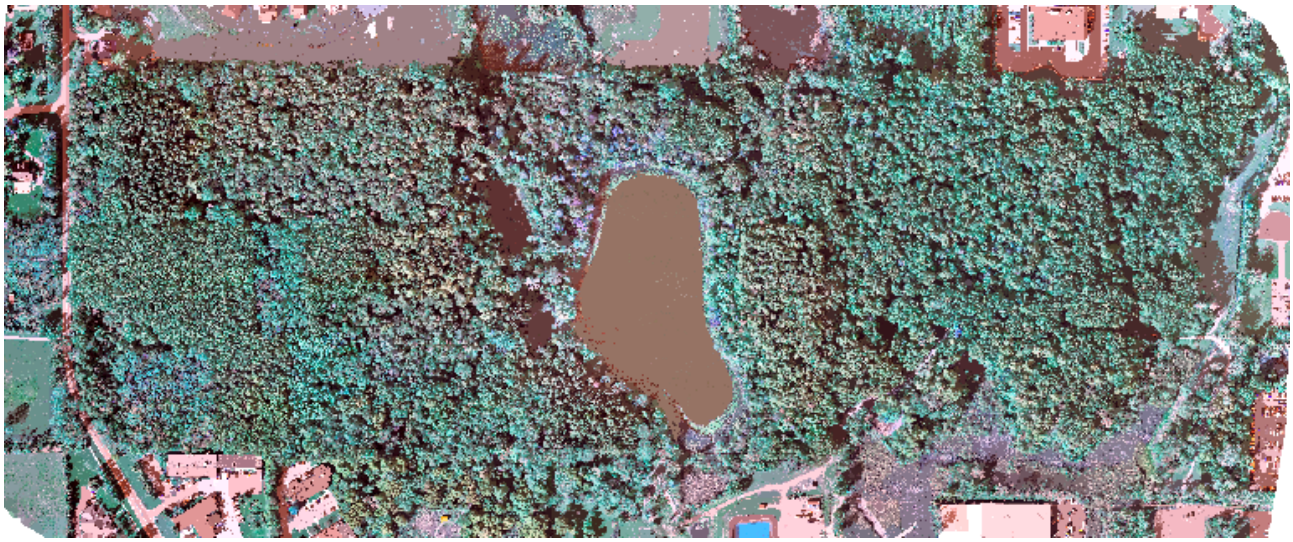
- Input raster: Select the segmented image.
- Training data: Confirm that the landcover community type samples are correctly set.
- Method: Choose an appropriate classifier (e.g., Random Forest, Support Vector Machine).

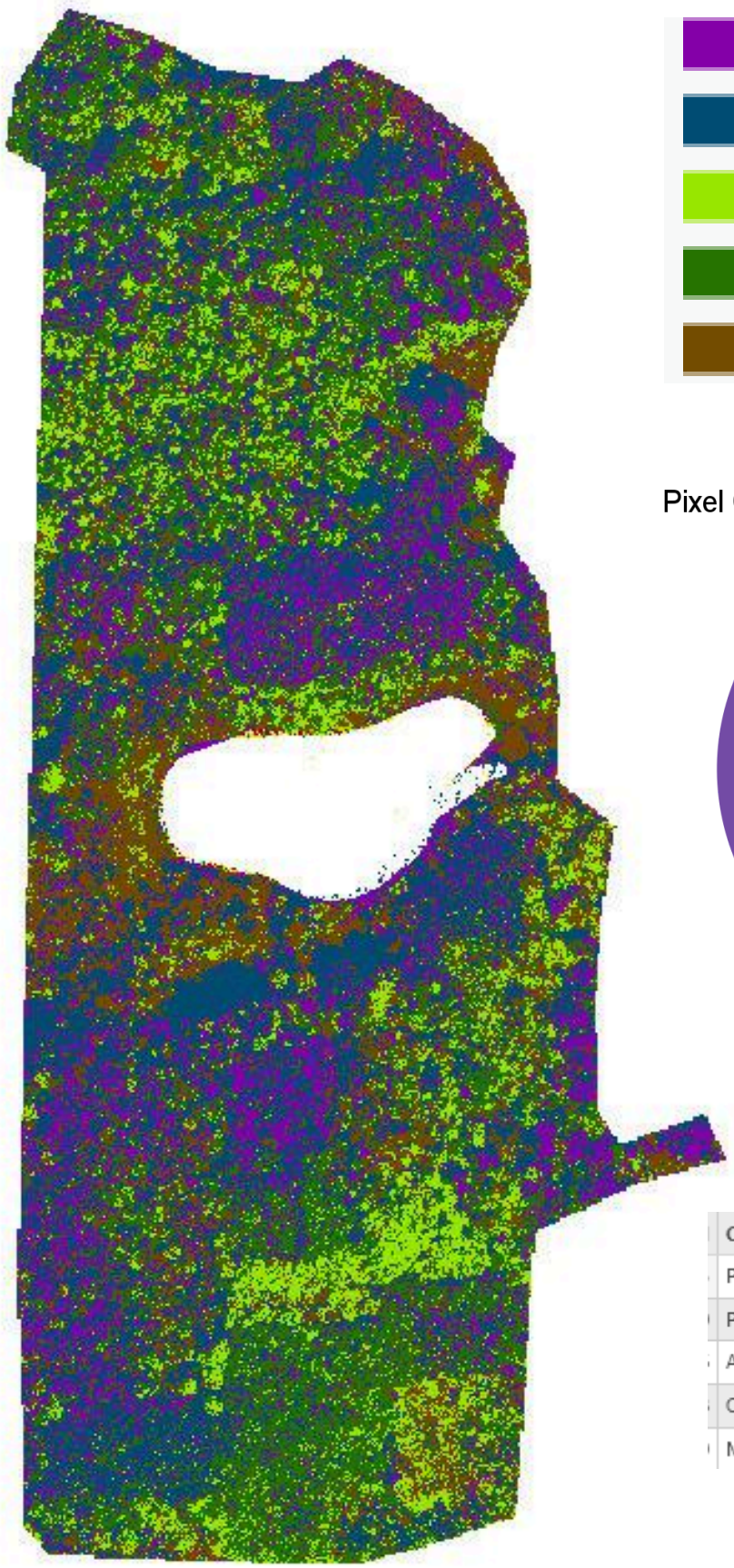
Enable Pixel Count Recording:

- Locate the checkbox labeled **Record Pixel Count** in the output options.
- Ensure this box is checked before proceeding.

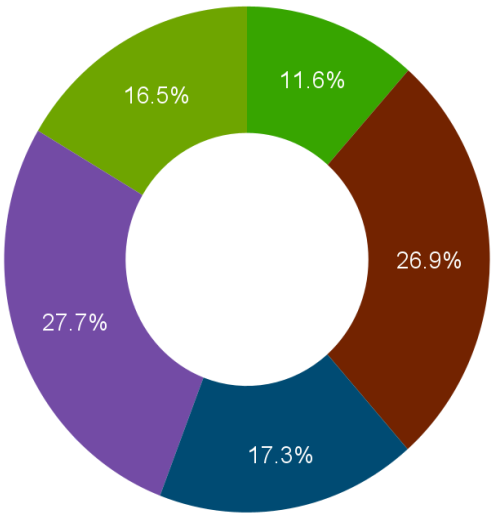
Click **Run** to execute the classification.







Pixel Composition



Class_name	Green	Blue	Count
Pinus strobus	255	255	7574
Picea abies	76	115	11356
Acer platanoides	168	0	3960
Quercus rubra	0	0	5599
Mixed Forest	230	0	10812

Allometric Equations

Jenkins and Ter-Mikaelian & Koruzukhin

Species	Jenkins	Ter-Mikaelian & Korzukhin
Acer platanoides	$M = 0.02 \cdot D^{2.43}$	$M = 0.04 \cdot D^{2.5}$
Picea abies	$M = 0.05 \cdot D^{2.28}$	$M = 0.07 \cdot D^{2.3}$
Pinus strobus	$M = 0.03 \cdot D^{2.37}$	Not provided
Quercus rubra	$M = 0.04 \cdot D^{2.36}$	$M = 0.06 \cdot D^{2.41}$

Tree height is provided by the canopy analysis

Combining Everything

What you can now do is use the canopy centers provided by pycrown and pair that with the image classification. Use these two to determine the approximate family level species of tree at a given canopy point. Use the DSM to figure out the height at that point.

Create a spreadsheet like:

dbh	height	species	Jenkings	Ter-Mikaelian & Korzukhin	Average
4.5	7.77	Acer.platanoides	0.7732793347	1.718269478	1.245774406
38.5	23.85	Quercus.rubra	205.9778248	310.2113744	258.0945996
35	35.7	Picea.abies	165.7467973	249.1463574	207.4465774
61	34.25	Pinus.strobus	510.9200708		510.9200708

Using the height generated from the pycrown layer to get individual points with a heigh attached. Use the overlap of the height and the image classification to add an attribute to the table that has the classification type that is present below the canopy center points. Use this to populate the table above

Calculate the biomass using the two equations and average the findings for each tree identified by the previous steps. Use ground truthed measurements to calculate the biomass for at least 3 individuals for each species. Use this to actualize the data calculated in the biomass column. Sum the total to get the total above ground biomass.

Accuracy Assessment and What Would I do Differently
Use ground based tree data to determine if the image classification for a given tree/area is accurate.

I would then use biomass estimates per acreage (https://www.researchgate.net/figure/Average-biomass-per-acre-tonnes-by-age-class-and-forest-type-for-the-Laskin-study-area_tbl2_238622225)
synthesized by multiple reports in order to determine the accuracy of the biomass calculation for the acreage of Saginaw Forest.

Something I would do differently is segment the image into areas that look to have unique canopy compositions. This would shorten the time it takes for pycrown to run and the separation of these forest types may give a greater understanding of the forest canopy heights. Segmenting the full NAIP image into sections would also shorten the image classification process.

This workflow was a great first step for me to estimate the biomass for a given section of forest using aerial imagery. A refined workflow would be able to produce a fairly accurate estimate with little ground truthed data. Restrictions to this project come with the scope and processing time/power required. Extrapolating canopy estimates as well as land classifications over large sections of forest is inefficient and the main changes moving forward would come from segmenting and reducing the workload for a given task.,