1. Appendices

1.1. Source code

Calculation of Leaf Area Index for Planet:

Ready.ipynb | sentinel-2.ipynb

Code

```
[1]: import numpy as np
     np.seterr(divide='ignore', invalid='ignore')
     import rasterio
     from rasterio.plot import show
     import matplotlib.pyplot as plt

[2]: # Define the input file path and load the image
     filepath = "/Users/akhrystodulov/lai/Planet_imagery/clip_planet.tif"
     image = rasterio.open(filepath)

[3]: # Read the red and near-infrared bands of the image
     red = image.read(3)
     nir = image.read(4)

     # Calculate NDVI using the red and near-infrared bands
     ndvi = np.divide(nir - red, nir + red)

     # Define the constant K for your vegetation type
     K = 1.0

     # Calculate LAI using the NDVI and K values
     LAI = K * ndvi

[11]: # Visualize the LAI
      plt.imshow(LAI, cmap='viridis')
      plt.colorbar()
      plt.title('Leaf Area Index from Planet')
      plt.show()
```

Fig 8.1.1

To calculate LAI for Planet satellite imagery conda environment was used with installed packages such as jupyter lab, numpy, rasterio, and matplotlib.

In the first steps, the necessary packages were imported. After that file path was defined. To open the Planet imagery rasterio package was used as it perfectly works with satellite imagery.

In the next steps we need to define red and nir bands to properly calculate the NDVI index, and based on it create LAI raster. The source code for the calculation LAI for Planet you can find at Fig 8.1.1.

Calculation of Leaf Area Index for Sentinel-2:

Ready.ipynb | sentinel-2.ipynb | Code

[23]:
```
import rasterio
import numpy as np
np.seterr(divide='ignore', invalid='ignore')
import matplotlib.pyplot as plt
```

[24]:
```
# Load the required bands
with rasterio.open('/Users/akhrystodulov/lai/s2/R10_B2.tif') as b2:
    band2 = b2.read(1)
with rasterio.open('/Users/akhrystodulov/lai/s2/R10_B3.tif') as b3:
    band3 = b3.read(1)
with rasterio.open('/Users/akhrystodulov/lai/s2/R10_B4.tif') as b4:
    band4 = b4.read(1)
with rasterio.open('/Users/akhrystodulov/lai/s2/R10_B8.tif') as b8:
    band8 = b8.read(1)
```

[25]:
```
# Create a numpy array for each band
red = np.array(band4, dtype=float)
nir = np.array(band8, dtype=float)
blue = np.array(band2, dtype=float)
green = np.array(band3, dtype=float)

# Calculate NDVI
ndvi = (nir - red) / (nir + red)

# Convert NDVI to LAI
lai = (ndvi / 0.69) ** 3.0

# Save the LAI as a GeoTIFF file
with rasterio.open('LAI.tif', 'w', driver='GTiff',
                   width=b2.width, height=b2.height,
                   count=1, dtype='float32',
                   crs=b2.crs, transform=b2.transform) as dst:
    dst.write(lai, 1)
```

[26]:
```
# Visualize the LAI
plt.imshow(lai, cmap='viridis')
plt.colorbar()
plt.title('Leaf Area Index from Sentinel-2')
plt.show()
```

Fig 8.1.2.

The LAI calculation process for Sentinel-2 is not completely different. As the common differences were described in the paper, the technical difference is only in the way of accessing bands, as they are stored in separate files. For LAI calculation NIR and RED bands were needed.

The source code below is responsible for the validation of the calculated LAI and provides statistical output such as a scatter plot of LAI values from Planet and Sentinel, and also shows the histogram of each raster. For the analysis, 2000 points were randomly specified from each raster which took a specific raster value. This approach avoids reshaping the dimensions of the data, while Planet and Sentinel have a different spatial resolution.

```
import rasterio
import numpy as np
import random
import matplotlib.pyplot as plt

# Load the Planet LAI raster
planet_lai =
rasterio.open('/Users/akhrystodulov/lai/Planet_imagery/lai.tif')
# Load the Sentinel LAI raster
sentinel_lai = rasterio.open('/Users/akhrystodulov/lai/s2/lai.tif')

# Define the number of random points to select
num_points = 2000

# Generate random row and column indices for the Planet LAI raster
planet_rows = np.random.randint(0, planet_lai.height, num_points)
planet_cols = np.random.randint(0, planet_lai.width, num_points)

# Generate random row and column indices for the Sentinel LAI raster
sentinel_rows = np.random.randint(0, sentinel_lai.height, num_points)
sentinel_cols = np.random.randint(0, sentinel_lai.width, num_points)

# Extract the LAI values at the random points from the Planet LAI raster
planet_lai_values = []
for row, col in zip(planet_rows, planet_cols):
    val = planet_lai.read(1, window=((row, row+1), (col, col+1)))
    planet_lai_values.append(val[0][0])

# Extract the LAI values at the random points from the Sentinel LAI
raster
sentinel_lai_values = []
for row, col in zip(sentinel_rows, sentinel_cols):
    val = sentinel_lai.read(1, window=((row, row+1), (col, col+1)))
    sentinel_lai_values.append(val[0][0])

rmse = np.sqrt(np.mean(np.square(np.subtract(planet_lai_values,
sentinel_lai_values))))
print(f'RMSE: {rmse}')

# Create a scatter plot of the LAI values from the two rasters
plt.scatter(planet_lai_values, sentinel_lai_values)
plt.xlabel('Planet LAI')
plt.ylabel('Sentinel LAI')
plt.title('Correlation between Planet and Sentinel LAI')
plt.show()

# Create histograms of the LAI values from the two rasters
plt.hist(planet_lai_values, bins=10)
plt.xlabel('LAI')
plt.ylabel('Frequency')
plt.title('Histogram of Planet LAI')
plt.show()

plt.hist(sentinel_lai_values, bins=10)
plt.xlabel('LAI')
plt.ylabel('Frequency')
plt.title('Histogram of Sentinel LAI')
plt.show()
```

```
# Calculate RMSE, MAE, and R-squared
diff = np.array(planet_lai_values) - np.array(sentinel_lai_values)
mse = np.mean(diff ** 2)
rmse = np.sqrt(mse)
mae = np.mean(np.abs(diff))
corr = np.corrcoef(planet_lai_values, sentinel_lai_values)[0, 1]
r_squared = corr ** 2

# Print the results
print("RMSE:", rmse)
print("MAE:", mae)
print("R-squared:", r_squared)
```

```
RMSE: 0.595954
MAE: 0.5161859
R-squared: 0.003342792050018555
```

1.2. User Manual

To use the LAI estimation tool, please follow these instructions:

1. Install the Conda environment

If you don't have Conda installed, please download and install the latest version of Miniconda from the official website: https://docs.conda.io/en/latest/miniconda.html

2. Open a terminal window and navigate to the project directory.

3. Create a new Conda environment using the following command:

conda env create -f environment.yml

4. Activate the Conda environment

Once the environment has been created, activate it using the following command:

conda activate lai-estimation

5. Install additional packages

The required packages are included in the environment.yml file, but if you need to install additional packages, use the following command:

conda install <package_name>

6. Run the tool

To run the tool, navigate to the project directory and execute the following command:

python lai_estimation_tool.py

7. Use the tool
Follow the prompts in the tool to input the required data and parameters.

The output will be saved to a file in the output directory.

If you encounter any issues or errors, please refer to the user manual or contact the author for assistance.

1.3. List of Abbreviations

LAI: Leaf Area Index

RS: Remote Sensing

NDVI: Normalized Difference Vegetation Index

SVM: Support Vector Machine

ANN: Artificial Neural Network

RNN: Recurrent Neural Network

CNN: Convolutional Neural Network

RMSE: Root Mean Squared Error

MAE: Mean Absolute Error

MSA: Mean Squared Error

SD: Standard Deviation

CV: Cross-Validation

ROI: Region of Interest

1.4. Glossary of Terms

LAI: Leaf Area Index. A measure of the amount of leaf area per unit of ground area.

Remote Sensing: The acquisition of information about an object or phenomenon without making physical contact with the object. In this study, remote sensing refers to the use of satellite data to estimate LAI values.

NDVI: Normalized Difference Vegetation Index. A commonly used vegetation index that is calculated using reflectance values from the red and near-infrared bands of remote sensing data.

Spatial Resolution: The level of detail in the imagery data expressed as the size of the smallest discernable feature. In this study, spatial resolution refers to the size of the pixels in the satellite data used to estimate LAI values.

Temporal Resolution: The time interval between two consecutive measurements of a given area. In this study, temporal resolution refers to the frequency of satellite data acquisition used to estimate LAI values.

Random Forest: A machine learning algorithm used for classification and regression tasks. In this study, Random Forest was used to estimate LAI values from satellite data.

Training Set: A set of data used to calibrate or train a machine learning algorithm. In this study, a training set was used to train the Random Forest algorithm to estimate LAI values from satellite data.

Validation Set: A set of data used to test the performance of a machine learning algorithm. In this study, a validation set was used to evaluate the accuracy of the Random Forest algorithm in estimating LAI values from satellite data.