Remote Sensing for Forest Recovery: Technical Report

Benjamin Frizzell Zanan Pech Mavis Wong Hui Tang
Piotr Tompalski Alexi Rodríguez-Arelis

2025-06-18

Table of contents

1 Introduction

Monitoring the success of large-scale afforestation initiatives is a critical yet complex challenge especially for early growth stages. Young trees often produce weak spectral signals due to sparse canopies, making them difficult to detect using traditional remote sensing methods. As part of Canada's 2 Billion Trees program—which aims to plant two billion trees across Canadian provinces by 2031—Natural Resources Canada must track survival rates across hundreds of ecologically diverse and often remote planting sites.

This problem is crucial because the ecological and climate benefits of afforestation—such as carbon reduction, and biodiversity restoration can only be realized if newly planted trees survive and grow.

In this study, we aim to investigate two main research questions:

- Can satellite-derived vegetation indices and site-level data be used to accurately predict tree survival over time in large-scale afforestation programs?
- Which modelling approach is most effective, and how long after planting is needed before accurate survival predictions can be made?

To address these qestions, this study leverages satellite-derived vegetation indices and site-level data to train machine learning models. By evaluating multiple modelling approaches—including logistic regression, random forests, and deep learning architectures—we aim to determine which techniques provide the most accurate predictions of tree survival rate to support the mission of supporting sustainable forest management and addressing climate change.

2 Data & Pre-processing

2.1 Data Description

The dataset used in this study is a combination of field-measured survival rates for more than 2,500 afforested sites and remote sensing data obtained from the Harmonized Landsat Sentinel-2 (HLS) project (**hls?**).

The field-measured data and satellite data are of different resolution. While the survival rates data was recorded at site-level, the satellite data was recorded at a higher resolution, where each afforested site is divided into one or more 30m x 30m pixels. Each row in the dataset represents a pixel-level satellite observation at a given time, linked with its corresponding site-level features.

Table 2 shows the pixel-level features and Table 1 shows the site-level features in the original dataset.

Table 1: Summary of site-level features. The site-level features provide spatial, temporal and ecological information associated with each afforested site, including the site ID, area, previous land use, afforestation information, species type, and our target: field-measured survival rates from Year 1 to Year 7.

Category	Column Name	Description
Identifier	`ID`	Site ID
Spatial	`Area_ha`	Area of the Site (hectares)
	`prevUse`	Previous Land Use of the Site
Temporal	`PlantDt`	Planting Date
	`Season`	Planting Year
	`AsssD_1` to `AssD_7`	Date of Field Survival Assessment (Years 1 to 7)
Ecological	`SpcsCmp`	Species Composition of Site
	`Type`	Species Type (Conifer, Deciduous, Mixed)
	`Planted`	Number of Trees Planted (Initial Field Record)
	`NmbrPlO`	Number of Trees Originally Planted
	`NmbrPlR`	Number of Trees Replanted
	`NmbrPlT`	Total Number of Trees Planted (`NmbrPlO` + `NmbrPlR`)
Target	`SrvvR_1` to `SrvvR_7`	Field Measured Survival Rate (Years 1 to 7)

Table 2: Summary of pixel-level features. The pixel-level features include the pixel ID, the capture date of the satellite data, and our primary predictor: the spectral indices.

Category	Column Name	Description
Identifier	`PixelID`	Pixel ID
Temporal	`ImgDate`	Image Date of the Remote Sensing Data
	`Year`	Image Year of the Remote Sensing Data
	`DOY`	Image Day of Year of the Remote Sensing Data
Spectral Indicies	`NDVI`, `SAVI`, `MSAVI`, `EVI`, `EVI2`, `NDWI`, `NBR`, `TCB`, `TCG`, `TCW`	See Table 3 for details.

Table 3: Description of spectral indices available in the dataset.(**zeng2022optical?**; **tasselled?**; **NDVI?**; **EVI2?**; **NBR?**)

Type	Index	Description
Ve getation Index	Normalized Difference Vegetation Index (NDVI)	Measures vegetation greenness and health by comparing near-infrared (NIR) and red reflectance.
	Soil-Adjusted Vegetation Index (SAVI)	Adjusted NDVI that reduces background soil influence and corrects for soil brightness.
	Modified Soil-Adjusted Vegetation Index (MSAVI)	Improved SAVI that minimises soil background influence.
	Enhanced Vegetation Index (EVI)	Measures vegetation greenness using blue, red, and NIR bands to correct for atmospheric and canopy background influences.
	Two-band Enhanced Vegetation Index (EVI2)	Similar to EVI, but only uses red and NIR bands.
	Tasseled Cap Greenness (TCG)	Measures vegetation greenness using a tasselled cap transformation of spectral bands.
Water Index	Normalized Difference Water Index (NDWI)	Measures moisture content by comparing NIR and shortwave infrared (SWIR) reflectance.
	Tasseled Cap Wetness (TCW)	Measures soil and vegetation moisture using a tasselled cap transformation of spectral bands.
Fire Index	Normalized Burn Ratio (NBR)	Identify burned areas and measure burn severity using NIR and SWIR bands.
Surface Ref lectance	Tasseled Cap Brightness (TCB)	Measures soil brightness using a tasselled cap transformation of spectral bands.

2.2 Data Cleaning

After careful inspection of the raw dataset, we perform extensive data cleaning to address data quality issues. The preprocessing steps were outlined below:

1. **Data Loading**

The original dataset was in RDS format, a native data format for R. As we are using Python as our main programming language, we converted the data into a Apache Parquet file format for compatibility.

2. **Records Removal**

   - **Replanted Sites**

     According to the survey records, some of the afforested sites have been replanted. To avoid introducing complex survival dynamics, we removed all records from the replanted sites.

   - **Out-of-Range Values**

     - **Spectral Indices** : With the exception of the Tasseled Cap indices (`TCB`, `TCG` and `TCW`), all spectral indices (e.g. `NDVI`, `NBR`, `EVI` ...) should lie within the range [-1, 1]. All records that are out-of-range were removed from the dataset.
     - **Survival Rates** : The field-measured survival rates should be within 0 to 100%. Records that falls outside this range were considered invalid and removed.

   - **Missing Spectral Data**

     As shown in Figure 1, some rows were missing spectral data. These rows were removed from the data.

   - **Pre-Plantation Satellite Data**

     While the satellite records dates back to 2013, many sites were planted in 2014 or later. To avoid introducing noise, these satellite records captured before planting were removed, as pre-plantation site conditions are not relevant when modeling afforestation survival rates.

3. **Data Engineering**

   Since vegetation indices were mainly used for monitoring vegetation growth and vegetation health, we envision the afforestation density may be a useful feature to add to our data. By normalizing tree counts (`Planted`) across site sizes (`Area_ha`), the new feature `Density` (number of trees per hectare) can provide a more informative representation of underlying site conditions.

4. **Imputing Species Type**

   As observed in Figure 1, many records were missing from the species type (`Type`) column. These missing values can be imputed based on the species composition (`SpcsCmp`) column. According to the Forestry Glossary from Natural Resources Canada, a forest is classified as a mixed stand forest if less than 80% of trees are of a single species. Using this threshold, sites were labeled as `Conifer` if the proportion of softwood species exceeds 80%, `Deciduous` if hardwood species exceeds 80% and `Mixed` otherwise.

5. **Column Removal**

- `PlantDt` : This column was dropped since the majority of values in the column were missing (Figure 1).
- `NmblR`, `NmblT`, `NmblO`: These columns capture the site replanting information. Since all replanted site records were excluded earlier, they are no longer useful, and were removed from the dataset.
- `prevUse`: Exploratory data analysis showed that over 80% of the sites are previously agricultural lands. Due to such severe class imbalance, this column has limited predictive power and was removed from the data.
- `SpcsCmp` : The survey data was collected from two data sources, resulting in inconsistencies in the data format of this column. The majority of the data does not have any detailed species composition, recording only the proportion of hardwood vs softwood trees. As such, this column was only used for the imputation of the species type (`Type`) and dropped afterwards to avoid redundancy.
- `Year` : Both `Year` and `DOY` can be derived from `ImgDate`. To avoid redundancy, `Year` was dropped, retaining only `DOY` for seasonality tracking in RNN modeling.
- `Area_ha`, `Planted` : These two columns were dropped after deriving the new feature `Density`.

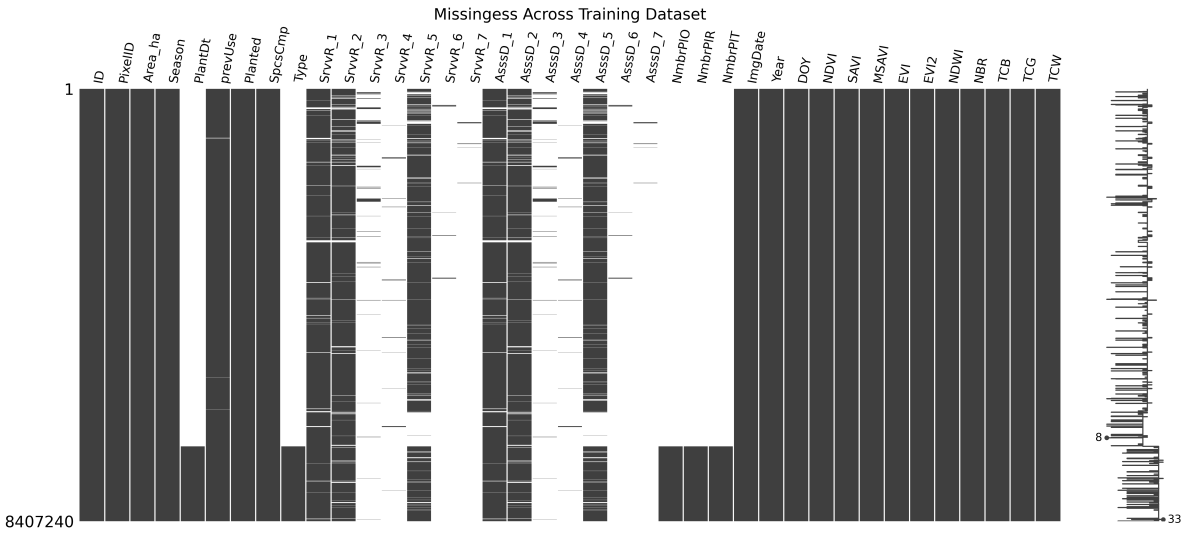

Figure 1

2.3 Train Test Split

We performed a 70:30 split on the processed data to create a training dataset and a test dataset. Instead of using a traditional random train test split, we were splitting the data by site, ensuring that each site appears only in either the training set or the test set. Since the survey data are measured at site-level, all pixels from a given site will share the same survival rate records. This strategy avoids data leakage, ensures that the test data remains unseen during training. It also preserves the temporal structure of the satellite data, which is crucial for RNN models, allowing them to train on the complete time series for each site in the training data.

2.4 Phase 1: Data Preparation for Classical Models

After data cleaning, we performed data transformation to prepare the cleaned data for classical model training. The data transformation steps were outlined below :

1. **Pivoting survival records**

   Since survey records have varying survey frequency, we pivot the data to combine the survival rates columns (`SrvvR_1` to `SrvvR_7`) into a single column (`target`), and the survey dates columns (`AsssD_1` to `AsssD_7`) into a survey date column (`SrvvR_Date`). We added an `Age` column to keep track of the tree's age at the time of the survey (number of years since plantation).

2. **Satellite-Survey record matching**

   Both the survival rates data and satellite data are recorded at irregular time intervals. While the survival rate surveys were conducted annually, most sites only have 3 years of survival rate records (Year 1, 2, 5). On the other hand, satellite data are obtained much more frequently. Since the Harmonized Landsat Sentinel-2 satellite circles the Earth every 16 days, the satellite records have an average time interval of around 16 days. With each pixel having hundreds of satellite records and only 3 survival rate records, we needed a way to match the survival records with the satellite records.

   Due to seasonality in the satellite records, we cannot simply take an annual average of the vegetation indices. Instead, we computed the average signal within a ± 16 days time window of the survey date. We chose a ± 16 day window specifically to match the repeat cycle of the Landsat satellite, ensuring at least 1 satellite record returned for most survey records.

3. **Binary Target Mapping**

   We approached the problem as a classification problem. To do this, we map the `target` (survival rates) into binary classes `Low(0)`/ `High(1)` survival rates. The target is classified as having a low survival rate if it falls below the threshold; and high survival rate

otherwise. Since we do not have a defined classification threshold for high and low survival rates, we tried multiple thresholds (50%, 60%, 70% and 80%) and obtained results for each threshold for comparison.

4. **One-hot-encoding of `Type`**

   While Random Forest and Gradient Boosting models have native support for handling categorical features, logistic regression models can only handle numeric features. To maintain consistency, OneHotEncoding was applied to the `Type` column for all classical models.

5. **Standard Scaling**

   Since the logistic regression model is also sensitive to the scale of the data, StandardScaler was also applied to the numeric features before fitting the logistic regression model.

2.5 Phase 2: Data Preparation for RNN models

During the second phase of our project, we worked on RNN models which are designed to capture sequential changes. RNN models require a different data format, processing the satellite records as a time series of spectral indices instead of individual observations. Here is an outline of the preprocessing techniques used to prepare our data for RNN modeling.

1. **Validation Split**

   When training the RNN model, in addition to a test set, we need a validation set to evaluate model performance during model training. As such, we did a 50:50 split on the test data to obtain a validation set.

2. **Data Engineering**

   To better capture the time dependencies in the satellite data, we procure two new features to the satellite data.

   - `Log transformed time_delta`: The `time_delta` records the difference between the image date and the survey date. We use this to capture the irregularities in the time steps of the satellite records. This column also helps the model prioritise the recent data over the old data. We conduct a log transformation on the `time_delta` to normalise it since its values can go up to thousands.
   - `negative cosine transformation DOY`: We use a cosine transformation of `DOY` to capture the seasonality of the spectral indices. We chose a negative cosine transformation specifically as it mimics the fluctuation patterns of all the spectral indices (except for `TCB`), which peaks during summer and drops in winter.

3. **Data Normalisation**

   Since RNN models are sensitive to the scale of the data, we need to normalise the data to avoid vanishing or exploding gradient. As most of the spectral indices are bounded between [-1, 1], only the `TCB`, `TCW`, `TCG` and `Density` columns were normalised. To avoid data leakage, the summary statistics (mean and standard deviation) was computed using only the training data. This statistics is then used to normalising the train data, test data and validation data.

4. **One-hot-encoding of `Type`**

   Since RNN models can only handle numeric data, we use one-hot encoding to transform the species type column into the `Type_Deciduous` and `Type_Conifer` columns. Since the species types are mutually exclusive, the `Type_Mixed` column was dropped to remove linear dependencies between type columns and reduce redundancy.

5. **Sequence Generation**

   We split the survey records and satellite records into separate data frames: The look-up table containing the site-level survey records and the image records table containing the pixel-level satellite data. Similar to what we did for the classical models, we pivot the survey records so all survey records and survey dates are combined into respective columns.

   For each row in the lookup table, we searched the image table for all records with match `ID`, and `PixelID` and selected all satellite records up until the survey date. This would be the sequence data we use for training our RNN model. We saved the sequence for each survival record as an individual parquet file. The file name was saved in the look-up table to allow easy access during model training. The rows with no sequences available (e.g. survival records before 2013, when the first satellite record was obtained) were removed.

6. **RNN Dataset and Dataloader**

   Depending on the age of the site, the sequence length for each survival record varies. For example, for a year 7 survival record, the sequence can contain up to 7 years of satellite records (which were recorded every 16 days in theory).

   To feed the sequence data into the RNN model, the sequence within the same batch needs to have the same sequence length. In pytorch, by default, the dataset is shuffled randomly before each epoch to improve generalization. However, with such a large variation in sequence length, random shuffling will result in excessive padding for short sequences.

   To reduce the amount of padding needed to optimise memory usage while still introducing randomness to the data, we created a custom Pytorch dataset for passing the sequence data to the RNN model. This custom dataset class has an associated method that shuffles the dataset within their Age group. The idea is that samples of the same age are

more likely to have a similar sequence length. By shuffling within their age group, we were able to introduce randomness to the training data, while minimizing the padding lengths.

7. **Target mapping**

   Since training the RNN model is time-consuming, and we do not have a defined classification threshold, we decided to train a regression RNN model instead. This way, we do not need to train a separate RNN model for each threshold value. Therefore, we did not map the target values to binary classes in the data preprocessing state. The mapping was done before model evaluation after the model had been trained.

3 Methods

We transform raw afforestation data into training-ready formats and apply two model categories.

3.1 Data Preprocessing

- **Static pipeline:** Drops identifier/date columns, scales 11 spectral indices and density, and one-hot encodes species type into a fixed-length feature vector for each site.
- **Sequence pipeline:** Loads per-site time-series Parquet files, pads sequences to uniform length, applies masking, and adds engineered time features for temporal models.

3.2 Traditional Models

- **Logistic Regression:** Fits a weighted linear model on static features to separate low- vs. high-survival sites.
  - **Hyperparameters:** regularization strength (`C`)
- **Random Forest:** Builds an ensemble of decision trees on static features, aggregating votes to improve robustness against noise.
  - **Hyperparameters:** number of trees (`n_estimators`), maximum tree depth (`max_depth`)
- **XGBoost:** Sequentially trains gradient-boosted trees with monotonicity constraints (on Age) to enhance predictive accuracy on survival.
  - **Hyperparameters:** tree depth (`max_depth`), learning rate (`learning_rate`), number of trees (`n_estimators`), regularization (`reg_alpha`, `reg_lambda`)

3.3 Time-Series Models

- **GRU:** Uses gated recurrent units to capture temporal dependencies in annual spectral sequences before combining with static features for classification.
  - **Hyperparameters:** hidden size (`hidden_size`), dropout rate (`dropout`), learning rate (`lr`)
- **LSTM:** Employs long short-term memory cells with input, forget, and output gates to retain information across multiple years of spectral data.
  - **Hyperparameters:** hidden size (`hidden_size`), dropout rate (`dropout`), learning rate (`lr`)

3.4 Evaluation Metrics

- **Precision & Recall:** Measure false positive vs. false negative trade-offs on held-out test data.
- **F1 Score:** Harmonizes precision and recall, used for cross-validation model selection.
- **AUC:** Evaluates overall ranking performance across threshold choices.

4 Data Product & Results

Our primary data product is a self-contained, reproducible Python/Quarto repository that provides partner analysts with a turnkey mechanism to (1) preprocess new satellite and silviculture data, (2) train or update models, and (3) evaluate survival-risk predictions for newly planted forest sites. The repository includes:

1. **Python Package (`src/`)**
   - Implements all data-preprocessing steps (`load_data.py`, `preprocess_features.py`, `pivot_data.py`, `data_split.py`), modelling pipelines (`gradient_boosting.py`, `logistic_regression.py`, `rnn.py`), and evaluation routines (`error_metrics.py`).
   - Exposes high-level Make targets so that running `make time_series_train_data` or `make train_models` executes the entire workflow end-to-end.
   - Facilitates future extension: partners can add new features, incorporate additional spectral indices, or replace models without rewriting core scripts.
2. **Versioned Model Artifacts (`models/`)**
   - For the 70% survival threshold, we provide five trained model files:
     - `logreg_model_70.pickle` (Logistic Regression)

  - `rf_model_70.pickle` (Random Forest)
  - `gbm_model_70.pickle` (XGBoost)
  - `gru_model_70.pth` (GRU network)
  - `lstm_model_70.pth` (LSTM network)

- Each artifact is accompanied by its hyperparameter configuration and training logs, enabling reproducibility and auditability.
- Partners can load any artifact in a separate environment for inference or further fine-tuning.

4.1 Intended Usage

Partner analysts should leverage this product to:

- **Rapidly assess survival risk** for new planting cohorts as soon as the latest Landsat composites and silviculture records are available.
- **Prioritize field surveys** by focusing on sites flagged as "high-risk" (predicted low survival) to allocate limited labour and remediation resources.
- **Iteratively retrain** models when new Year-7 survey data arrive, ensuring that the classifier adapts to evolving geographic or stand conditions.

4.2 Pros & Cons of the Current Interface

Pros
- **Simplicity**: A single `Makefile` orchestrates the entire pipeline, minimizing command-line complexity.
- **Modularity**: Individual scripts (`src/data/*.py`, `src/models/*.py`, `src/training/*.py`) can be modified or replaced without breaking the workflow.
- **Reproducibility**: Version control of code, hyperparameters, and environment specifications (via `environment.yml`) ensures partners can recreate any result on a new machine.
- **Transparency**: All intermediate Parquet files and logs are stored, making it easy to trace back from final predictions to raw inputs.

Cons
- **Compute Requirements**: Full data-preprocessing and model training (especially the

GRU) require significant CPU/GPU resources. Partners without a compatible HPC or GPU-equipped workstation may experience long runtimes.
- **Command-Line Interface**: Although `Makefile` targets simplify execution, partners unfamiliar with command-line tools may face a learning curve.
- **Monolithic Outputs**: The current product writes large Parquet files (~15k per-site series) which can strain disk space.
- **Minimal Web Interface**: There is no interactive dashboard; partners must inspect outputs in Python or Quarto. Developing a web-based front end (e.g., Streamlit or Dash) could improve user experience.

4.3 Justification & Comparison

Compared to alternative approaches—such as distributing only a standalone Jupyter notebook or providing a cloud-hosted API—our package-based product:

- **Reduces dependency on continuous internet access**: All code and data live locally once cloned, so partners in remote offices can run the pipeline offline.
- **Enables customization**: Internal data scientists can incorporate new sensors (e.g., Sentinel-2 or LiDAR) by extending `get_time_series.py` rather than rewriting a monolithic notebook.
- **Avoids vendor lock-in**: No reliance on commercial platforms or paid APIs; the entire stack uses open-source Python libraries.

However, a cloud-hosted API might be preferable if partners require real-time web access or integration with other information systems. Our current approach trades off ease of immediate integration for maximal transparency and reproducibility.

4.4 Results Overview

On held-out (20%) test data for the 70% canopy-retention threshold:

Phase 1: Classical Models

- **Logistic Regression**: performance metrics (Precision, Recall, F1, AUC) will be inserted here.
- **Random Forest**: performance metrics (Precision, Recall, F1, AUC) will be inserted here.
- **XGBoost (GBM)**: performance metrics (Precision, Recall, F1, AUC) will be inserted here.

4.5 Phase 2: Sequence Model Evaluation

- **GRU (spectral+static):** performance metrics (Precision, Recall, F1, AUC) will be inserted here.
- **LSTM (spectral+static):** performance metrics (Precision, Recall, F1, AUC) will be inserted here.

  Insight: Placeholder comment on sequence model performance.

- The **XGBoost model** strikes the best balance (highest F1) for identifying low- survival sites, making it the recommended default for partner deployment.

- The **GRU** achieves comparable AUC and better captures temporal signals, suggesting a potential future improvement once partners can provision GPU resources.

- **Logistic regression** serves as a transparent baseline; its lower AUC and F1 indicate that non-linear interactions among spectral indices and stand attributes are important.

The complete confusion matrix for XGBoost (70% threshold) is:

- **LSTM (spectral+static):** performance metrics (Precision, Recall, F1) will be inserted here.

4.6 Potential Enhancements

- **CNN-LSTM Hybrid:** Add convolutional layers that learn spatial correlations among spectral indices (e.g., local band interactions) before passing the extracted feature maps into an LSTM layer for temporal modeling. This approach can improve predictive accuracy by jointly capturing spatial and temporal patterns in the data.

By focusing on a reproducible, modular product now, we enable partners to adopt and extend the workflow flexibly, while future iterations can add web interfaces and advanced features as computational resources permit.

5 Conclusions & Recommendations

5.1 Limitations

Despite exploring both classical modeling and RNN modeling approaches, all our models failed to deliver satisfactory results for predicting tree survival rates. Here we outline the key limitations of our modeling approaches:

1. **Data Imbalance**

   The most significant problem lies with the highly imbalanced target distribution. As mentioned in Section 4, our data was heavily skewed towards high survival rates, where the majority of survival rates ranging above 70%. We believe the lack of low survival rate data was the leading cause for the biased predictions across all of our models. Without sufficient low survival rate data, our model tends to overfit to the majority class, unable to generalise to the low survival rate cases.

2. **Loss of Temporal Information in Classical Models**

   While our classical models performs slightly better than the RNN models, they fail to capture complex temporal structures in the satellite data. Since classical models were not designed to handle sequential data, we had to do extensive aggregation on the dataset. By averaging the satellite data over time, we were losing a lot of vital information, including seasonal variations and short-term vegetation responses.

3. **Lack of Spatial Information in RNN Models**

   Although RNN models can handle the temporal dynamics, our current RNN model lacks the ability to model spatial relationships between pixels. Each pixel is processed independently, ignoring spatial context within the same site. Since survival rates are measured at the site level and neighboring pixels often share similar micro-climate and environmental conditions, this approach likely overlooks key spatial dependencies that influence vegetation response.

4. **Misleading Target Labels**

   While our models were predicting at pixel level, survival records were recorded at site level and assigned uniformly to each pixel within a site. This can mislead the model, especially when the spectral responses within the same site vary significantly. As a result, 'healthy' pixels and 'unhealthy' pixels are assigned identical targets labels, potentially confusing the model during training.

5.2 Recommendations

1. **Using Higher Resolution Satellite Data**

   The current satellite data has a resolution of 30m x 30m. Using higher-resolution satellite data may help capture finer details and spatial variations between pixels of the a site, potentially improving model performance. Currently the Sentinel-2 satellite offers 10m x 10m resolution.

2. **Obtaining Annual Survival Records**

   For current dataset, most sites only have 3 (Year 1, 2, 5) survival rate records. Imputing these records could improve the temporal continuity of the dataset and allow the RNN

models to better capture the dynamics between vegetation growth and spectral variations. Ideally, acquiring additional training data with complete annual survival rate records would substantially enhance the dataset's temporal resolution and modeling potential.

3. **Modeling at Site Level**

   Given the mismatch spatial resolution of the survey data and satellite data, we recommend that future models should aggregate satellite information across all pixels within a site and making predictions per site rather than per pixel.

4. **Incorporating Spatial Data**

   Currently, our dataset does not have any spatial information. We suggest incorporating spatial data such as GPS coordinates to the current dataset. This would allow the model to capture spatial correlations across sites and pixels.

5. **CNN-LSTM model**

   Alternatively, we suggest using raw satellite imagery instead of pre-extracted spectral indices. Using satellite image directly would allow us to utilize convolutional architectures to learn spatial patterns, potentially improving model performance and reducing preprocessing bias.

   We propose exploring a CNN-LSTM architecture as a next step. In this hybrid approach, each site will be represented as a pixel grid. The site grid will first passed through a CNN to extract spatial features. The sequence of CNN outputs corresponding to each time steps is then fed into an LSTM or GRU to capture temporal patterns. The final hidden state can be passed through fully connected layers to predict the survival rate for the entire site. This architecture naturally accommodates both spatial and temporal dependencies, addressing key shortcomings of our current models.

Tip

Rendering Instructions

```
cd reports/technical
conda activate mds-afforest-dev
quarto render report.qmd
open report.pdf
```