Using Random Forest to Classify Landscape Disturbances in OLYM, MORA, and NOCA

Standard Operating Procedure

Dan Wexler, John Boetsch, and Natasha Antonova

4/19/2024

----------------------------------------------------------------------------------------------------

Table of Contents

1. Introduction
   1. Overview
   2. Getting Started
2. Assembling Predictor Variables
   1. ArcGIS Pro
   2. Google Earth Engine
3. Variable Selection
4. Model Training
5. Labeling
6. Quality Assurance / Quality Control
   1. Method
   2. Future
7. Appendix
   1. Adding Predictors
   2. Default Variable Values

----------------------------------------------------------------------------------------------------

Document Description

This standard operating procedure (SOP) details the step-by-step process of creating a random forest machine learning model and using it to classify landscape disturbance patches in Olympic, Mount Rainier, and North Cascades National Parks. It is focused mostly on *how* to create and run the model, not *why* the model operates in a certain way. For more information about the reasoning behind the process, see the report.

1 Introduction

1.a Overview

In partnership with Oregon State University, the North Coast and Cascades Network employs LandTrendr, an algorithm that uses Landsat satellite imagery to detect landscape change, to find landscape disturbances in Pacific Northwest National Parks. This algorithm identifies pixels that have experienced change and groups neighboring pixels from the same year into disturbance patches. While these patches indicate where disturbance has likely occurred, the type of disturbance (e.g., avalanche, blowdown, etc.) remains unknown. The goal of this procedure is to build a machine learning model that can accurately classify patches as one of a discrete number of disturbance types. The workflow covered by this SOP begins after LandTrendr.

The following steps outline this process at a high level and indicate the software used at each step. The different sections of this document describe the steps in more detail.

1. Generate geometric predictor variables (ArcGIS Pro)
2. Generate remaining predictor variables (Google Earth Engine)
3. Select predictor variables to include in the model (R)
4. Train a random forest model on labeled data and assess its accuracy (R)
5. Use the trained model to label unseen disturbance patches (R)
6. Perform quality assurance / quality control: validation and final accuracy assessment on a subset of the new labels (ArcGIS Pro)

1.b Getting Started

The user will need access to ArcGIS Pro, the Google Earth Engine (GEE) Code Editor, and RStudio in order to carry out this procedure. ArcGIS Pro and GEE are used to calculate predictor variables for the disturbance patches, and RStudio is used to train and test the model. ArcGIS Pro will also be used for QAQC after labeling.

To begin, download the [Landscape-Disturbance-Classification GitHub repository](https://github.com/NPS-NCCN/Landscape-Disturbance-Classification) to a local machine either by cloning it using Git or downloading it as a zip file. Within the resulting workspace, there are four scripts that will guide the user through this modeling process, shown in Table 1. The first script will need to be copied and pasted into the GEE Code Editor, and the remaining three scripts are R Markdown files that can be opened in RStudio. The user should ensure that they can open these scripts in their respective applications before proceeding.

*Table 1: Descriptions of the code scripts that are used to build the model.*

|  |  |
| --- | --- |
| **Script** | **Description** |
| gee\_generate\_predictors.txt | Google Earth Engine script that generates predictor variables for disturbance patches |
| rf\_variable\_selection.Rmd | R Markdown script that assists the user in removing correlated predictor variables |
| rf\_training.Rmd | R Markdown script that trains a random forest model on labeled data |
| rf\_labeling.Rmd | R Markdown script that labels unseen data using a trained random forest |

In addition to these scripts, the workspace contains folders for each park (OLYM, MORA and NOCA), and each park folder contains five sub-folders that control the flow of data throughout the modeling process. These five folders are detailed in Table 2. The first and fourth folders, “1\_training\_input” and “4\_labeling\_input”, are the only folders that require user input. After running the labeling script, the labels for the unseen disturbance patches will appear in the “5\_output” folder. Note that there is a file in each sub-folder titled “ignore.txt”. These files can be ignored but not deleted, as they help preserve the file structure of the project when pushing and pulling from GitHub.

*Table 2: Descriptions of the folders that handle the flow of data throughout modeling.*

|  |  |
| --- | --- |
| **Folder** | **Description** |
| 1\_training\_input | The user uploads predictor variable information for disturbance patches that are labeled |
| 2\_starting\_variables | Contains the variables that the random forest will consider when modeling, generated using “rf\_variable\_selection.Rmd” |
| 3\_intermediate | Contains a trained random forest model, the list of variables used to train the model, and two files with model probability information, generated using “rf\_training.Rmd” |
| 4\_labeling\_input | The user uploads predictor variable information for disturbance patches that are unlabeled |
| 5\_output | Contains the labels generated by the model for the unseen disturbance patches |

Finally, there is a folder in Microsoft Teams…

*NCCN-Landscape-RS/Documents/General/Dan/Final/gee\_assets*

…that contains assets that will need to be uploaded to GEE. Download these files now.

2 Assembling Predictor Variables

2.a ArcGIS Pro

The user will now upload the disturbance patches generated by LandTrendr to ArcGIS Pro and use the Zonal Geometry Tool in the Spatial Analyst Toolbox to calculate the geometric predictor variables. The user will also add the columns “In\_Park”, “In\_Buffer”, “In\_Mask”, and “Protected” to the data, which all provide information on where in a park a disturbance patch is located. These columns are used to filter the data throughout the modeling process.

2.b Google Earth Engine

At this point, the user should have a collection of four files that together contain the LandTrendr disturbance patches with geometric predictor variables added on. This collection should include a .dbf, .prj, .shp and .shx file. The user will now feed these patches into GEE to calculate the remaining predictor variables.

To register for GEE and create a project, navigate to this link…

*https://code.earthengine.google.com/register*

…and follow the steps. You may have to register for a Google Cloud account.

In the GEE Code Editor, notice the three main panels (Figure 1). On the left, there is a panel with the “Scripts”, “Docs”, and “Assets” tabs. “Scripts” is where code lives, and “Assets” is where the user will upload the GEE assets from the Microsoft Teams folder. The middle panel is where the user will paste “gee\_generate\_predictors.txt”. The right panel contains the “Inspector”, “Console”, and “Tasks” tabs. The user will mostly interact with the “Tasks” tab. The following steps detail this part of the protocol.

1. Download the “gee\_generate\_predictors.txt” file from the GitHub repository, then navigate to the GEE Code Editor and open the “Scripts” tab on the left panel. Click “NEW -> File” and copy and paste the contents of “generate\_predictors.txt” into the new GEE file using the middle panel.
2. Open the “gee\_assets” folder in the Microsoft Teams (Figure 2). This folder contains images and shapefiles that the GEE script will use to calculate predictor variable information for each disturbance patch. There should be 10 folders in total, with three folders for each park and an extra folder for NOCA. Upload each asset to GEE by navigating to the “Assets” tab in the GEE Code Editor and clicking the “New” button. If the asset is a shapefile, click the “Shape file” button. If the asset is an image, click the “GeoTIFF” button. Click “SELECT” and upload all the files that are associated with a single asset. Ensure that each file is named as described in Table 3, substituting in the four-letter name of the park for “PARK”.

*Table 3: Information about the GEE assets for each park.*

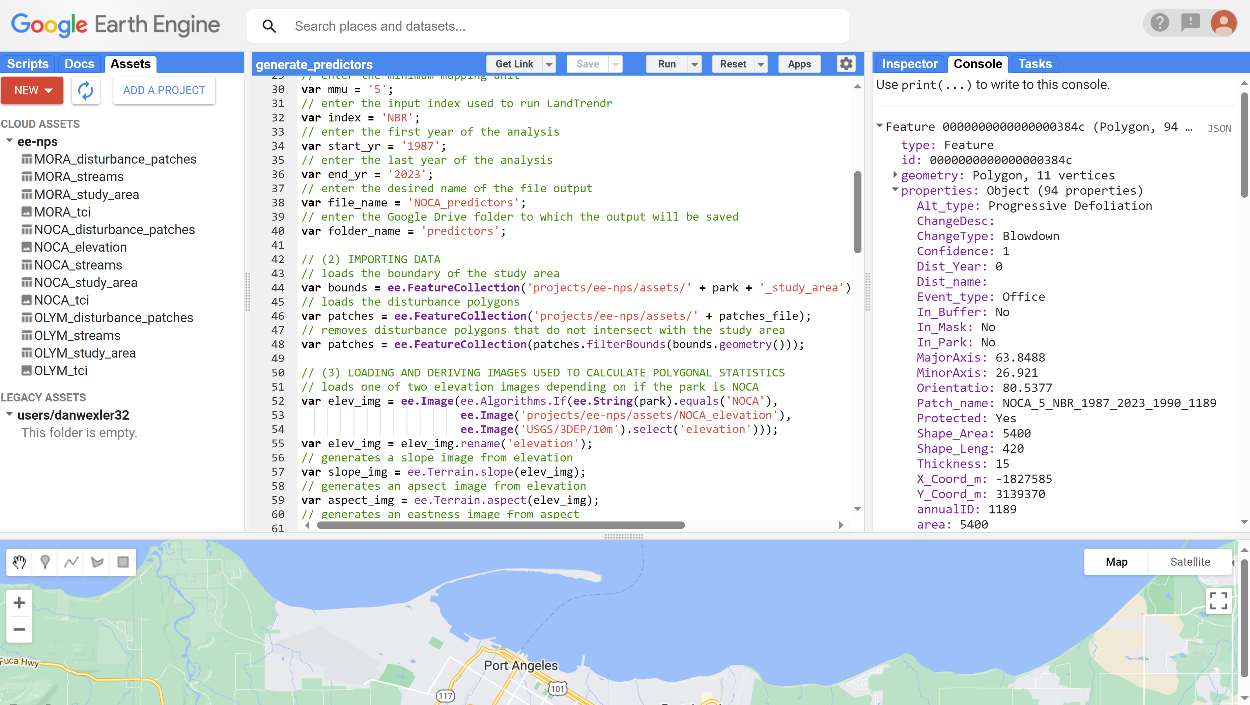
|  |  |  |
| --- | --- | --- |
| **Asset** | **Description** | **File Type** |
| PARK\_boundary | Bounds of the study area | shape |
| PARK\_streams | Streams throughout the park | shape |
| PARK\_tci | Topographic convergence index image | image |
| NOCA\_elevation | Elevation image for NOCA | image |

1. Upload the disturbance patches shapefile to GEE as an asset using the method described in the previous step. Choose a descriptive name for the patches.
2. Create a folder in Google Drive using the account that is associated with GEE. This is the folder to which the results of the GEE script will save.
3. Section one of “gee\_generate\_predictors.txt” contains a few variables that should be modified by the user. Modify these variables now by changing the text and saving the file using the “Save” button at the top of the middle panel.

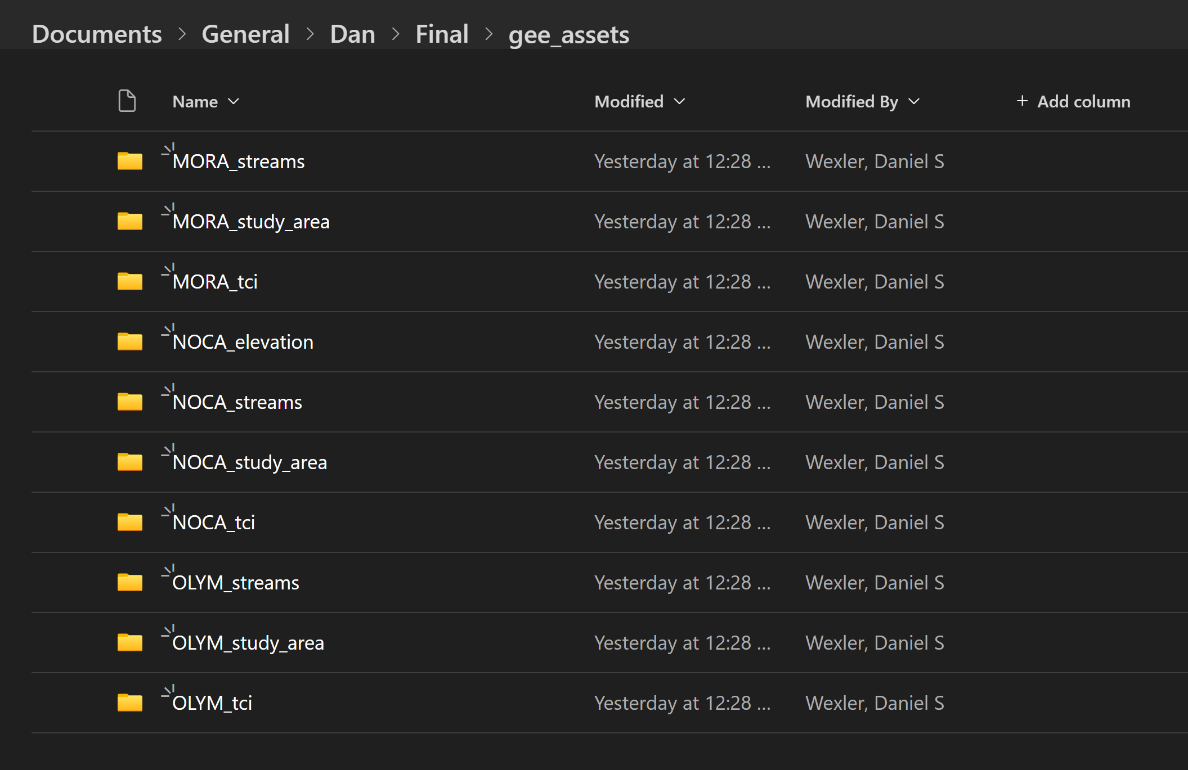
*Table 4: A list of variables the user should modify before running the GEE script.*

|  |  |
| --- | --- |
| **Variable** | **Description** |
| park | The four-letter name of the park that is being modeled |
| patches\_file | The name of the GEE asset containing the disturbance patches |
| mmu | The minimum mapping unit of the LandTrendr run |
| index | The spectral index used to identify change in the LandTrendr run |
| start\_yr | The start year of the LandTrendr run |
| end\_yr | The end year of the LandTrendr run |
| file\_name | The name of the output .csv file |
| folder\_name | The name of the Google Drive folder where the output will save |

1. Finally, execute the script by clicking the “Run” button at the top of the middle panel, and notice that the “Tasks” tab on the right panel lights up. Navigate to this tab and click the “RUN” button next to the name of the task. A window will pop up with the name of the task, the output folder, the output file name, and the file format. Ensure these fields are correct and click “RUN” one final time. Expect multiple hours of runtime (runs have gone as long as 14 hours, so check back in the morning!). Once the script is finished, a .csv file containing the predictor variable information will appear in the specified Google Drive folder. If the user is training a random forest model, they should proceed to Section 3 of this document, which details the pre-training variable selection process. If the user is labeling new disturbance patches, they should proceed to Section 5.



*Figure 1: The Google Earth Engine Code Editor interface. The right panel contains the user-uploaded assets, and the middle panel contains code from “gee\_generate\_predictors.txt”.*



*Figure 2: The Microsoft Teams folder that contains the Google Earth Engine assets. Each park has a streams layer shape file, a study area shape file, and a topographic convergence index image. NOCA has an additional elevation image.*

3 Variable Selection

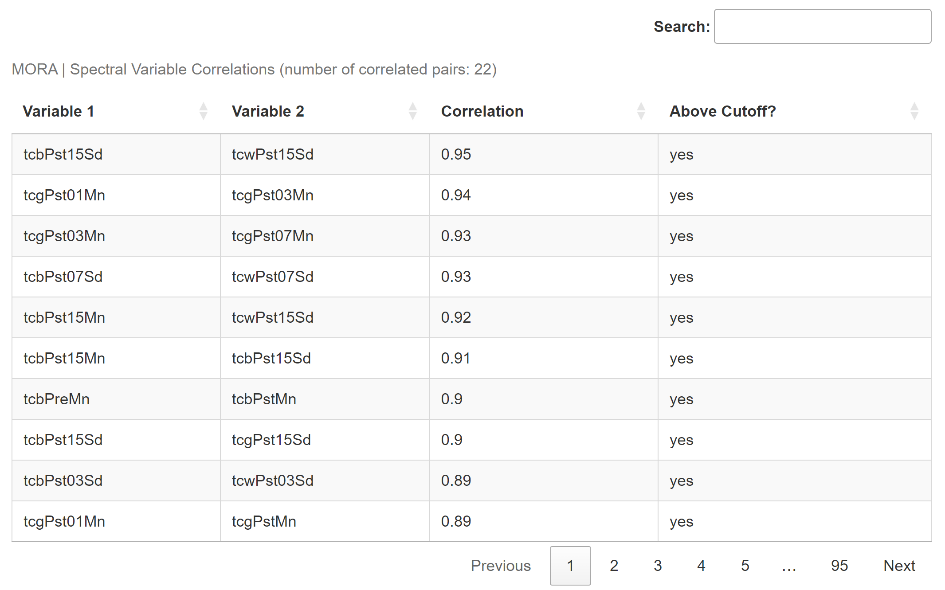
This section of the procedure details how to use the “rf\_variable\_selection.Rmd” script to choose the set of predictor variables that will be available to the random forest model by reducing or eliminating redundant (highly correlated) variables. The script provides tools for visualizing correlations between variables and allows the user to flexibly include or exclude any variables they choose. The following steps detail how to proceed through this part of the protocol.

1. Open the “Landscape-Disturbance-Classification” project in RStudio by clicking “File -> Open Project” and selecting the .Rproj file in the GitHub repository. Then, click on “File -> Open File” and open “rf\_variable\_selection.Rmd”.
2. Upload the output of the GEE script to the “1\_training\_input” folder for the current park. Ensure the file is named something unique and understandable.
3. The second code block in “rf\_variable\_selection.Rmd” contains a few variables that should be modified by the user. Modify those variables now. On the initial run of this script (it will be run multiple times), it is recommended that all the variables in the “spec”, “land”, “geom” and “spti” dictionaries are set to “T” to include all variables in the initial set of correlation matrices.

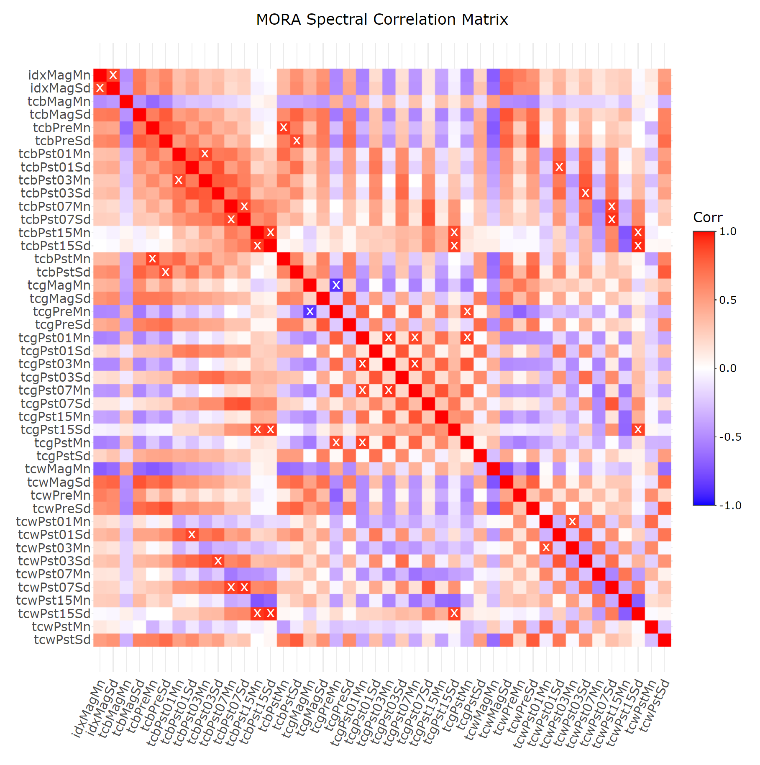
*Table 5: A list of variables the user should modify before running the correlation script.*

|  |  |
| --- | --- |
| **Variable** | **Description** |
| park | The four-letter name of the park that is being modeled |
| predictors\_file | The name of the .csv file containing predictor variables |
| correlation\_cutoff | A value between 0 and 1 that indicates the correlation coefficient cutoff above which pairs of variables are flagged |
| filter\_out\_mask | Filter out disturbances within the elevation mask? |
| drop\_1987 | Filter out disturbances that occurred in 1987, the first year of the time series? |
| model\_pa\_and\_buffer | Filter patches to those within the protected areas + buffer? |
| model\_park\_and\_buffer | Filter patches to those within the park boundary + buffer? |
| model\_pa | Filter patches to those within the protected areas? |
| model\_park | Filter patches to those within the park boundary? |
| spec | A list of spectral predictors that can be turned on or off |
| land | A list of landscape predictors that can be turned on or off |
| geom | A list of geometric predictors that can be turned on or off |
| Spti | A list of space/time predictors that can be turned on or off |

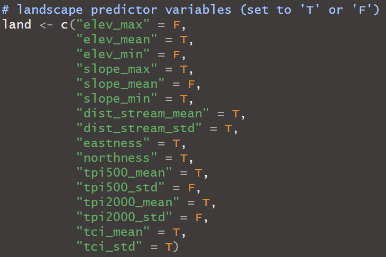
1. Run the script by clicking “Knit” at the top of RStudio. Open the resulting “rf\_variable\_selection.html” file by navigating to the “html\_files” folder in the GitHub repository, or view the output in RStudio using the “Viewer” tab. Go to the “(2) Categorical Correlation” section. For each category of predictor variable (spectral, landscape, geometric, and space/time), there are two tools that will help the user identify those that are correlated. The first tool is a table that contains every pair of predictor variables from the current category (switch between the categories by clicking on different tabs at the top of the section) and their Spearman rank correlation coefficient (Figure 3). The table is ordered according to the magnitude of the correlation coefficient, so the values at the top have the highest positive and negative correlations. The user can switch between pages using the numbers at the bottom of the table and is able to search for specific variables using the bar at the top. The “Above Cutoff?” column contains a “yes” or a “no” depending on whether that pair has a correlation above the user-defined cutoff. The second tool is a correlation matrix, which is simply a different way of visualizing the table. The “x” marks indicate pairs of variables with a correlation above the user’s cutoff (Figure 4). Use the table and the matrix to identify correlated variable pairs. Note that the “(3) Final Correlation” section contains a table that compares variables across categories. Look at this table to ensure that all correlated pairs of variables are at least recognized.
2. Make a list of variables that, if removed from their respective categories, would reduce the number of correlations. For a pair of correlated variables, how should the user decide which to remove? That is ultimately up to the user, but it is recommended that the user removes the least interpretable of the two variables. For example, if the mean brightness of the patch one month after the disturbance is correlated with the standard deviation of the brightness, it would make sense to remove the standard deviation and keep the mean, as standard deviation is less interpretable. Because these predictors are correlated (in this example), removing either variable should not affect the model significantly.
3. Go back to the top of the script and set the variables to remove to “F” (Figure 5). Rerun the script and look at the correlation table and matrices. Are there still correlated pairs? If so, the user can repeat this process by setting more variables to “F” and rerunning the script. The user should continue running the script until they are happy with the list of variables set to “T”. These are the variables that will be available to the random forest model during training and will be saved to the “2\_starting\_variables” folder.



*Figure 3: A table showing pairs of correlated variables that the user can use to assist the variable selection process. The correlation coefficient for each pair is shown, as well as a variable indicating whether that correlation is above the user-defined cutoff.*



*Figure 4: A correlation matrix showing pairs of correlated variables that the user can use to assist the variable selection process. The “x” marks indicate pairs of variables with a correlation above the user-defined cutoff.*



*Figure 5: A list of variables that the user will modify to choose which variables to keep and which to remove. The user should set variables they wish to keep to “T” and those they wish to remove to “F”.*

4 Model Training

This section details how to use “rf\_training.Rmd” to train a random forest model. Before proceeding with this section, ensure that predictor variables have been assembled (Section 2) and that variables have been selected (Section 3). The following steps detail how to proceed through this part of the protocol.

1. Ensure that there is already a file containing predictor variables in the “1\_training\_input” folder. For an example of what this file should look like, navigate to the following folder in Microsoft Teams:

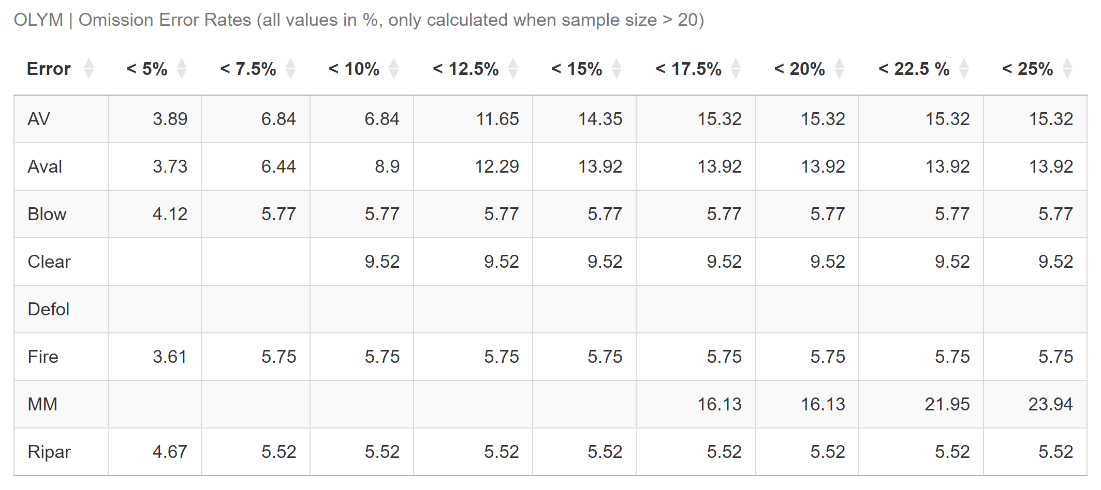
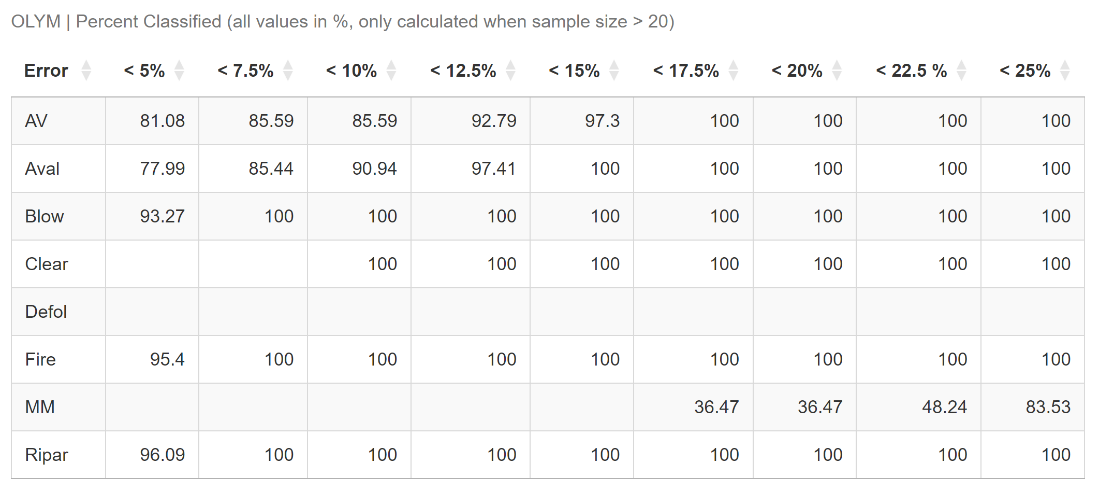
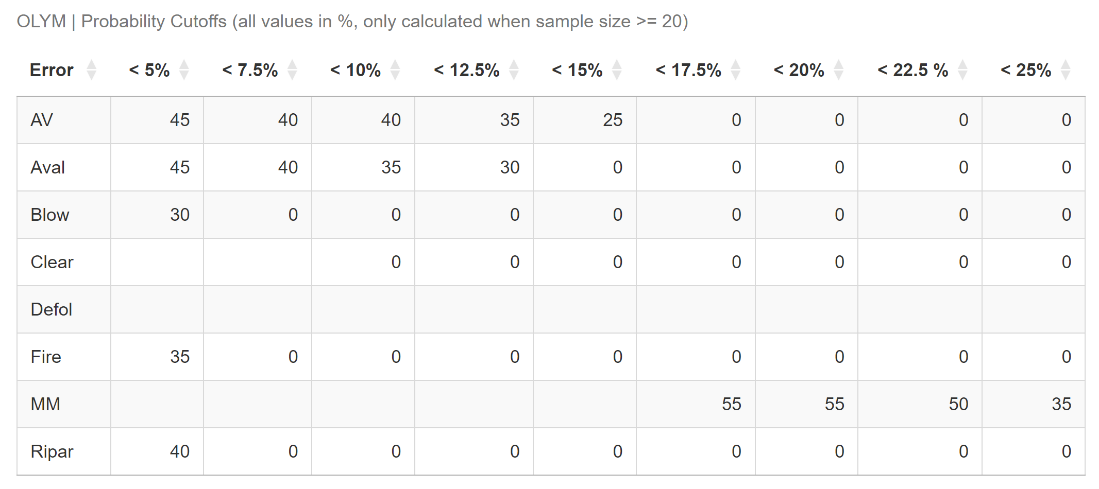
*NCCN-Landscape-RS/Documents/General/Dan/Final/patches\_for\_training*

1. In addition to the columns containing predictor variable information, make sure that the file contains the following fields:
   1. ChangeType: the disturbance type, can be blank.
   2. In\_Mask: whether the patch centroid is in the vegetation mask.
   3. In\_Buffer: whether the patch centroid is in the buffer.
   4. In\_Park: whether the patch centroid is in the park boundary.
   5. Protected: whether the patch centroid is in a protected area.
2. The second code block in “rf\_training.Rmd” contains variables that should be modified by the user. Modify those variables now.

*Table 6: A list of variables the user should modify before running the training script.*

|  |  |
| --- | --- |
| **Variable** | **Description** |
| park | The four-letter name of the park that is being modeled |
| predictors\_file | The name of the .csv file containing predictor variables |
| disturbance\_subset | A vector of numbers where each number represents a disturbance type to include in modeling process |
| filter\_out\_mask | Filter out patches fully within the vegetation mask? |
| drop\_1987 | Filter out disturbances that occurred in 1987, the first year of the time series? |
| model\_pa\_and\_buffer | Filter patches to those within the protected areas + buffer? |
| model\_park\_and\_buffer | Filter patches to those within the park boundary + buffer? |
| model\_pa | Filter patches to those within the protected areas? |
| model\_park | Filter patches to those within the park boundary? |
| group\_post\_avalanche | Relabel post-avalanche patches as avalanches? |
| group\_post\_blowdown | Relabel post-blowdown patches as blowdowns? |
| group\_post\_clearing | Relabel post-clearing patches as clearings? |
| group\_post\_fire | Relabel post-fire patches as fires? |
| group\_post\_mm | Relabel post-mass movement patches as mass movements? |
| group\_water\_with\_av | Relabel water patches as annual variability? |
| balance\_testing\_data | Balance the testing sets across disturbances? |
| train\_test\_split | The percent of the data for each disturbance type that will be used to train the model (between 0 and 1) |
| balance\_multiplier | The number of times larger the training set limit is than the number of examples of the smallest disturbance class |
| min\_train\_distance | The minimum allowed distance, in meters, between patches of the same disturbance type in the training set |
| num\_trees | The number of trees with which to build the random forest |
| num\_vars\_add | The number of top variables to add each time a new random forest is trained |
| train\_all\_data | Train a new model without withholding testing data once model probability cutoffs have been determined? |

1. Run the script by clicking “Knit”. Open the resulting “rf\_training.html” file by navigating to the “html\_files” folder in the GitHub repository or view the output in Rstudio using the “Viewer” tab. Read through the output to learn more about the training process, how the model performed on the testing set, and more. The trained random forest model and the list of variables included in the model will be saved to the “3\_intermediate” folder.
2. Go to the “(6) Probability Evaluation” section. The user will use the information in this section to decide, for each disturbance class, what model probability cutoff best balances omission error and the percent of patches classified (Figure 6). The text in this section contains detailed information about model probabilities that the user should read to better understand how to pick cutoffs. Decide on a model probability cutoff for each disturbance class. To elaborate on this decision-making process, imagine an example with the avalanche disturbance type in which a 50% model probability cutoff results in an omission error rate of 4% with 70% of the patches receiving accepted labels. At a 40% model probability cutoff, the error rate is 6.5% and the percent of accepted labels is 97%. Should the user choose the 40% or 50% model probability cutoff? The user must decide whether an expected increase in omission error of 2.5% is worth a 27% increase in the number of accepted labels. The user will have to make these sorts of calculations when deciding on probability cutoffs for each disturbance type.



*Figure 6: The three tables in the “(6) Probability Evaluation” of “rf\_training.Rmd” that the user will use to decide on a probability cutoff for each disturbance class. The top chart shows the model probability cutoffs at which each disturbance types achieved below certain omission error rates. The middle chart shows the percent of the patches that received accepted labels at the cutoffs shown in the first chart. The bottom chart shows the actual omission error rates for each disturbance type at the cutoffs shown in the first chart. The user should use all three of these charts together to decide on individual cutoffs.*

5 Labeling

The user can now apply the trained random forest model (Section 4) to unseen disturbance patches using the “rf\_labeling.Rmd” script. The following steps detail how to proceed through this part of the protocol.

1. Follow the steps in Section 2 of this document to assemble the predictor variables for the new LandTrendr disturbance patches. Put the .csv output of the GEE script in the “4\_labeling\_input” folder. For an example of what this file should look like, navigate to the following folder in Microsoft Teams:

*NCCN-Landscape-RS/Documents/General/Dan/Final/patches\_to\_label*

1. The second code block in “rf\_labeling.Rmd” contains a few variables that should be modified by the user. Modify those variables now. For the “probability\_cutoffs” variable, set the cutoff for each disturbance type to the value found while training the model in Step 4 of Section 4 (Figure 7). If the user is not modeling for a certain disturbance type, or if there is no suitable model probability cutoff for a specific disturbance type, set the cutoff to “NA”.

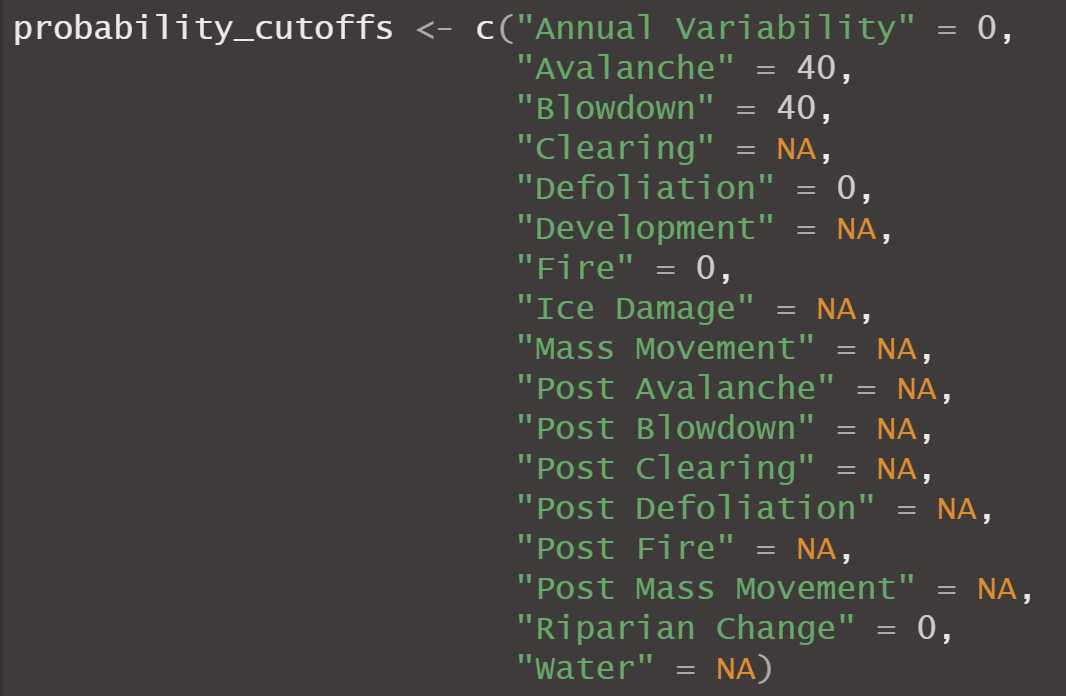
*Table 7: A list of variables the user should modify before running the labeling script.*

|  |  |
| --- | --- |
| **Variable** | **Description** |
| park | The four-letter name of the park that is being modeled |
| predictors\_file | The name of the .csv file containing predictor variables |
| save\_name | The desired name of the output file containing labels |
| Filter\_out\_mask | Filter out patches fully within the vegetation mask? |
| drop\_1987 | Filter out disturbances that occurred in 1987, the first year in the time series? |
| label\_pa\_and\_buffer | Filter patches to those within the protected areas + buffer? |
| label\_park\_and\_buffer | Filter patches to those within the park boundary + buffer? |
| label\_pa | Filter patches to those within the protected areas? |
| label\_park | Filter patches to those within the park boundary? |
| sample\_size | The number of patches of each disturbance type to flag for QAQC after labeling |
| probability\_cutoffs | A list of all disturbance types. Set the model probability cutoff for each disturbance class |

1. Run the script by clicking “Knit”. Open the resulting “rf\_labeling.html” file by navigating to the “html\_files” folder in the GitHub repository or view the output in RStudio using the “Viewer” tab. Read through the knitted html output to learn more about the labeling process, view a sample of the resulting labels, and see the distribution of labels across the different disturbance types.
2. Go to the “(4) Analysis” section. This section tells the user, based on the probability cutoffs that they set, the percent of the new disturbance patches that received an accepted label. The table in this section displays the distribution of labels across disturbance types, what percent of the labels for each disturbance class were accepted, and more. If the user is not satisfied with the percent of patches that received an accepted label, they may consider lowering the probability cutoffs and rerunning the script.
3. Open the output file that is now in “5\_output” (Figure 8). This file contains the labels for the unseen disturbance patches, as well as some additional information. Table 8 describes each column of the output.

*Table 8: Descriptions of the columns in the output label file*

|  |  |
| --- | --- |
| **Column** | **Description** |
| ID | A unique id for each disturbance patch |
| Label1 | The winning label assigned to the patch by the model |
| Prob1 | The percent of trees that voted for the winning label |
| Accept1 | Is the winning label acceptable based on the probability cutoffs set by the user? |
| Label2 | The second-best label assigned to the patch by the model |
| Prob2 | The percent of trees that voted for the second-best label |
| Accept2 | Is the second-best label acceptable based on the probability cutoffs set by the user? |
| QAQC | Should this patch be looked at for QAQC? The QAQC patches are randomly chosen within each disturbance type |

**

*Figure 7: A list of all the disturbance types and their user-chosen probability cutoffs. The disturbances with “NA” values are either not being modeled or never achieved an acceptable omission error rate and thus have no model probability cutoff.*

**

*Figure 8: A preview of the labeling output.*

6 Quality Assurance / Quality Control

6.a Method

While the omission error rates found by applying the random forest to the testing data are likely representative of how the model would perform on new, unseen data, it is good practice to perform an additional QAQC to ensure that this is the case. To facilitate this QAQC, the “rf\_labeling.Rmd” script flags a certain number of patches labeled as each disturbance type for review. The user will look at these flagged patches in ArcGIS Pro and record whether the label generated by the model is accurate. With this information, the user can calculate classification error rates for each disturbance type within each park, as well as generally for each model. The user can also use the secondary classifications to assist their hand-labeling decisions.

6.b Future

The user should train a new model for each park approximately every three years and use the current model to label new disturbance patches each year as those patches become available. Every time labels are generated for a set of unseen disturbance patches, a QAQC should be performed on those disturbance patches and error rates for each disturbance type and park should be calculated. When the current three-year period is up, the user can incorporate the patches that were hand-labeled labeled for QAQC into the next model for training. It is not advisable to include labels generated by the model in training, as the user cannot say with certainty that they are correct.

Congratulations on making it through this SOP!

Please email [danwexler32@gmail.com](mailto:danwexler32@gmail.com) with any questions.

7 Appendix

7.a Adding Predictors

This section walks through the steps of adding a new predictor variable to the model.

1. If the predictor variable can be calculated in ArcGIS Pro, compute the variable alongside the calculation of geometric predictor variables (Section 2).
2. If the predictor can be calculated in GEE, open “gee\_generate\_predictors.txt” in the GEE Code Editor and navigate to Section 3 of the code. This section loads GEE images and user-uploaded assets to create images that will be sampled using the disturbance polygons. Follow the examples of loading and creating images in that section and add the necessary code.
3. Navigate to Section 5 of the code and add a line of code for calculating the new predictor. The code should follow the format below. The image name is the name of the image variable created by the user, the reducer specifies whether to calculate the minimum, maximum, mean, or other aspect of the image, the field name is the output name of the variable field, and the band name is the name of the band in image that is going to be sampled.

Patches = patches.map(calc(IMAGE NAME, REDUCER, FIELD NAME, BAND)

1. If a variable is being created that does not sample an image, but instead uses other predictors to calculate a new predictor, follow the example of the “paratio” function in Section 4 of the code and write a new function. Then, add a line in Section 5 that follows the format below.

Patches = patches.map(NAME OF FUNCTION)

1. Click “Run” at the top of the middle panel and then navigate to the “Console” tab on the left panel. The script prints out the values of the predictor variables for the first disturbance patch so the user can ensure that predictors are being calculated correctly without running the entire script. Ensure that the new variable is being calculated correctly then run the script using the steps in Section 2.b of this SOP.
2. Finally, open “rf\_variable\_selection.Rmd” in RStudio, and add the name of the variable to one of the lists in the second code block, either “spec”, “land”, “geom”, or “spti”. The user can now incorporate the new variable into the modeling process.

7.b Default Variable Values

This section provides default values for the variables that the user can modify in each of four scripts. Some explanation on why a variable is a certain value is also provided where it is necessary. The user is encouraged to experiment with the variables to tune the models for each park.

*Table 9: A list of default variables for the “gee\_generate\_predictors.txt” script. Substitute “PARK” in the “patches\_file” and “file\_name” variables with the name of the park.*

|  |  |
| --- | --- |
| **Variable** | **Default** |
| park | “MORA”, “OLYM” or “NOCA” |
| patches\_file | “PARK\_disturbance\_patches” |
| mmu | “5” |
| index | “NBR” |
| start\_yr | “1987” |
| end\_yr | “2023” |
| file\_name | “PARK\_predictors” |
| folder\_name | “predictors” |

*Table 10: A list of default variables for the “rf\_variable\_selection.Rmd” script.*

|  |  |
| --- | --- |
| **Variable** | **Default** |
| park | “MORA”, “OLYM” or “NOCA” |
| predictors\_file | “predictors.csv” |
| correlation\_cutoff | 0.85 -> (captures the highly correlated variables) |
| filter\_out\_mask | FALSE -> (model is good at labeling patches in mask) |
| drop\_1987 | TRUE -> (the start year of a LandTrendr run is wonky) |
| model\_pa\_and\_buffer | TRUE |
| model\_park\_and\_buffer | FALSE |
| model\_pa | FALSE |
| model\_park | FALSE |
| spec | All T, initially |
| land | All T, initially |
| geom | All T, initially |
| Spti | All T, initially |

*Table 11: A list of default variables for the “rf\_training.Rmd” script.*

|  |  |
| --- | --- |
| **Variable** | **Default** |
| park | “MORA”, “OLYM” or “NOCA” |
| predictors\_file | predictors.csv |
| disturbance\_subset | c(1, 2, 3, 4, 5, 7, 9, 16) -> (disturbances that model well) |
| filter\_out\_mask | FALSE -> (model is good at labeling patches in mask) |
| drop\_1987 | TRUE -> (the start year of a LandTrendr run is wonky) |
| model\_pa\_and\_buffer | TRUE |
| model\_park\_and\_buffer | FALSE |
| model\_pa | FALSE |
| model\_park | FALSE |
| group\_post\_avalanche | TRUE -> (not enough data on its own to model) |
| group\_post\_blowdown | TRUE -> (not enough data on its own to model) |
| group\_post\_clearing | TRUE -> (not enough data on its own to model) |
| group\_post\_fire | TRUE -> (fire and post-fire are hard to distinguish) |
| group\_post\_mm | TRUE -> (not enough data on its own to model) |
| group\_water\_with\_av | TRUE -> (disturbances on water are false change) |
| balance\_testing\_data | FALSE -> (gives more accurate omission error rates) |
| train\_test\_split | 0.7 -> (normal value for train-test split) |
| balance\_multiplier | 10 -> (ensures there is enough data for larger classes) |
| min\_train\_distance | 200 -> (arbitrarily chosen, should be examined) |
| num\_trees | 1000 -> (arbitrarily chosen, should be examined) |
| num\_vars\_add | 5 -> (small enough to find a parsimonious model) |
| train\_all\_data | FALSE |

*Table 12: A list of default variables for the “rf\_labeling.Rmd” script.*

|  |  |
| --- | --- |
| **Variable** | **Default** |
| park | “MORA”, “OLYM” or “NOCA” |
| predictors\_file | predictors.csv |
| save\_name | disturbance\_labels.csv |
| filter\_out\_mask | FALSE -> (model is good at labeling patches in mask) |
| drop\_1987 | TRUE -> (the start year of a LandTrendr run is wonky) |
| label\_pa\_and\_buffer | TRUE |
| label\_park\_and\_buffer | FALSE |
| label\_pa | FALSE |
| label\_park | FALSE |
| sample\_size | 50 -> (arbitrarily chosen, seems like a good number) |
| probability\_cutoffs | NO DEFAULT, SET BASED ON MODEL |