Remote Sensing of Defensible Space Compliance to Enhance Wildfire Preparedness

Technical Documentation

A capstone project submitted in partial satisfaction of the requirements for the degree
of Master of Environmental Data Science
for the
Bren School of Environmental Science & Management

By
Joshua Cohen
Ryan Green
Leilanie Rubinstein
Rachel Swick

Committee in charge:
Mark Buntaine, Ph.D.
Carmen Galaz García, Ph.D.

May 2025

Clients:
Sarah Anderson, Ph.D.
Mark Buntaine, Ph.D.
Cesar Martinez-Alvarez, Ph.D

Bren School of Environmental Science and Management
Department of Political Science

SIGNATURES

Remote Sensing of Defensible Space Compliance to Enhance Wildfire Preparedness

As developers of this Capstone Project documentation, we archive this documentation on the Bren School's website such that the results of our research are available for all to read. Our signatures on the document signify our joint responsibility to fulfill the archiving standards set by the Bren School of Environmental Science & Management.

_____

Joshua Cohen

_____

Ryan Green

_____

Leilanie Rubinstein

_____

Rachel Swick

The Bren School of Environmental Science & Management produces professionals with unrivaled training in environmental science and management who will devote their unique skills to the diagnosis, assessment, mitigation, prevention, and remedy of the environmental problems of today and the future. A guiding principle of the School is that the analysis of environmental problems requires quantitative training in more than one discipline and an awareness of the physical, biological, social, political, and economic consequences that arise from scientific or technological decisions.

The Capstone Project is required of all students in the Master of Environmental Data Science (MEDS) Program. The project is a six-month-long activity in which small groups of students contribute to data science practices, products or analyses that address a challenge or need related to a specific environmental issue. This MEDS Capstone Project Technical Documentation is authored by MEDS students and has been reviewed and approved by:

_____

Dr. Mark Buntaine (faculty advisor)

_____

Dr. Carmen Galaz García (EDS 411 Instructor)

_____

Date Signed

ACKNOWLEDGMENTS

This project could not have been completed without the support and guidance of our faculty and capstone advisors, as well as our clients and other individuals.

Faculty Advisor
Dr. Mark Buntaine, Bren School of Environmental Science & Management

Capstone Advisor
Dr. Carmen Galaz García, Bren School of Environmental Science & Management

Clients
Dr. Sarah Anderson, Bren School of Environmental Science & Management
Dr. Mark Buntaine, Bren School of Environmental Science & Management
Dr. Cesar Martinez-Alvarez, UCSB Department of Political Science
Santa Barbara County Fire Department

Special Thanks
Cullen Molitor, Center for Effective Global Action & Environmental Markets Lab
Jon Jablonski, UCSB Library, DREAM Lab
Scott Safechuck, Public Information Officer, Santa Barbara County Fire Department

ABSTRACT

Climate change has fueled an increase in wildfire frequency and intensity in California, having profound impacts on communities across the state. The Santa Barbara County Fire Department (SBCFD) currently enforces defensible space regulations, which require homeowners to maintain a buffer zone around their homes free of combustible vegetation. These regulations mitigate the spread of wildfire to structures and allow firefighters to more safely defend homes. This project investigated the feasibility of using remotely sensed satellite imagery from Planet Labs in a machine learning model to determine property compliance with defensible space regulations. Our team determined that, at the imagery resolution currently available to us, our model can not predict compliance at a reliable rate. In addition, because the fire department may not have full access to inspected properties, we suspect that some homes are mistakenly marked as compliant based on only the portion of the property visible from the street. Next steps for improving the model include using a training set with higher resolution imagery, manually classifying images as compliant and non-compliant, and continuing to add and remove variables from the training data. We also recommend the fire department provide explicit reasons for why a property was marked as compliant or non-compliant during their inspections. We hope these insights will provide researchers with the next steps to create a defensible space machine learning model and help SBCFD with future inspection efforts.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

As wildfires become more common in the face of climate change, wildfire preparedness in wildland-urban interface zones is increasingly critical for community safety. Dead brush, trees, and other vegetation surrounding structures can fuel wildfires, allowing fire to jump from vegetation to human structures. Creating a defensible space involves removing, or trimming vegetation and other combustible fuels in the area around a structure to mitigate the spread of wildfire. Defensible space also provides firefighters with the clearance to defend properties from wildfire more safely and effectively.

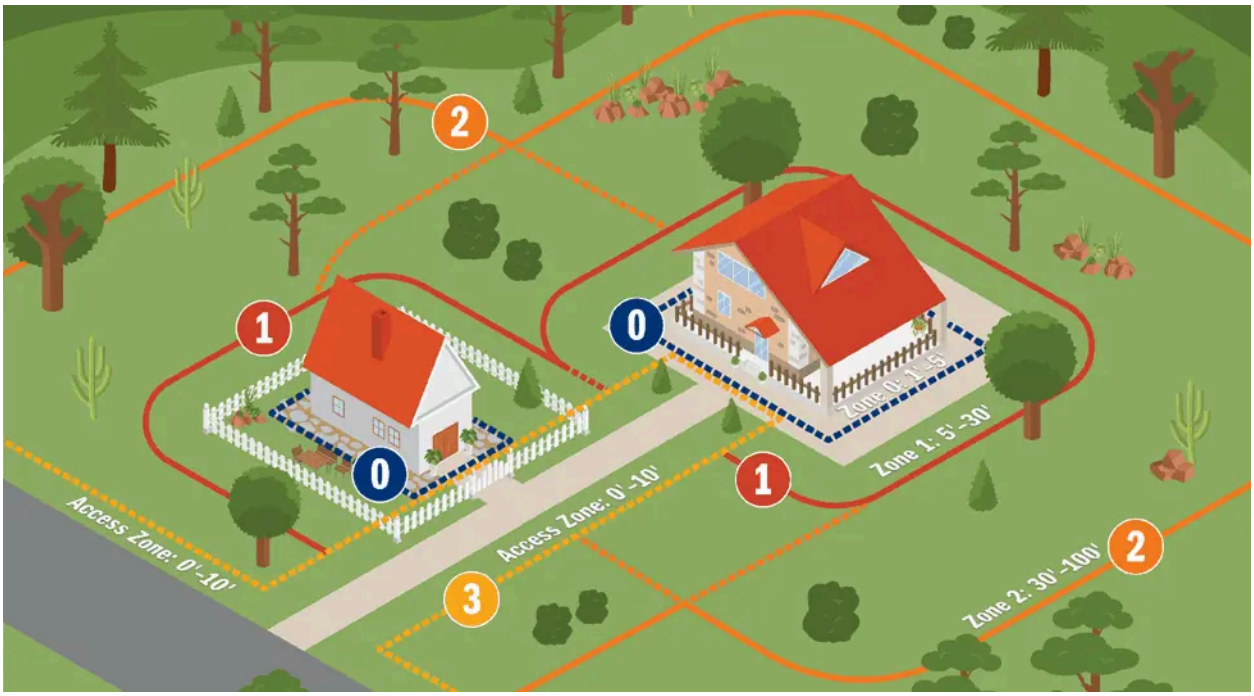

Figure 1: Defensible Space Zones
There are 3 defensible space zones around a structure: Zone 0, 0 - 5 ft, no ground vegetation or overhanging tree limbs; Zone 1, 0 - 30 ft, remove dead vegetation and keep vegetation trimmed; and Zone 2, 30 - 100 ft, maintain annual grasses at no more than 4 inches, and create space between grasses, shrubs, and trees. This is not a complete description of each zone's defensible space requirements. For a complete list please visit https://sbfiresafecouncil.org/defensible-space/. For this project we are only concerned with Zones 1 and 2. 2022 Santa Barbara County Fire Department, Defensible Space Diagram

The state of California mandates that all areas under state jurisdiction be classified into Fire Hazard Severity Zones of Moderate, High, or Very High. Since defensible space is an effective and cost efficient way to reduce risk to firefighters and potential damage to structures, the state of California has implemented defensible space requirements for residents who live in the Very High Fire Hazard Severity Zone of the Local Responsibility Area. These requirements include a 30-foot buffer around a structure free of any overhanging vegetation, and a 100-foot buffer free of any dead or dense vegetation (*Fig. 1*). In California, any building designed to house people, animals, or property over 120 square feet is considered a structure. Local fire departments are

tasked with ensuring all structures within their jurisdiction are compliant with state guidelines on defensible space.

Our project focuses on Santa Barbara County where the Santa Barbara County Fire Department (SBCFD) is tasked with conducting annual field inspections to ensure every structure within their jurisdiction is compliant with state defensible space regulations. However, these inspections are time-consuming and resource intensive as they require the dispatch of SBCFD personnel across the county, inspecting thousands of structures.

Motivated by increasing fire dangers, an interdisciplinary team of researchers at the Bren School of Environmental Science and Management wanted to determine how effective mailing information on defensible space guidelines to property owners living within fire prone areas of Santa Barbara County would be on increasing rates of defensible space compliance. They also hoped increasing compliance rates would decrease the enforcement workload for SBCFD. They have enlisted our help with the construction of a machine learning model that would identify compliance with defensible space regulations at the property level.

This project had two main objectives:

1. Use remote sensing imagery in a machine learning model to identify areas within Santa Barbara County with high levels of non-compliance with defensible space regulations.

2. Use the predictions from the machine learning model to increase the efficiency of SBCFD annual inspections.

To achieve these objectives, our team built a predictive classification model. A predictive classification model is a type of machine learning model that uses data with known classifications (in this case, Compliant or Non-Compliant) to learn patterns and relationships within that data. Once trained, it can accurately classify new data it has not seen before.

The SBCFD provided us with defensible space inspection data from 2018 to 2023, which we used to train a random forest model to predict defensible space compliance. A random forest model is a type of machine learning model composed of many decision trees. A decision tree is an algorithm in which a computer repeatedly asks questions about how to categorize its given data into two categories. Much like how a tree branches out, it then recursively splits the data into subsets with similar qualities until reaching some end condition. The resulting model is able to determine whether a given row in the data, in this case a property, is either compliant or non-compliant.

Our team determined that, at the imagery resolution currently available to us, our model can not predict compliance at a reliable rate. In addition, because the fire department may not have full access to inspected properties, we suspect that some homes are mistakenly marked as compliant based on only the portion of the property visible from the street.

Next steps for improving the model include using a training set with higher resolution imagery, manually classifying images as compliant and non-compliant, and continuing to add and remove variables from the training data. We also recommend the fire department provide explicit

reasons for why a property was marked as compliant or non-compliant during their inspections. We hope these insights will provide researchers with the next steps to create a defensible space machine learning model and help SBCFD with future inspection efforts.

I. APPROACH

This project was completed in three main phases: 1) Data Preparation, 2) Model Creation, and 3) Reproducible Pipeline. These three phases are outlined in Figure 2 below.

The Data Preparation phase involved accessing, cleaning, and storing the data used in the project, including inspections data from the Santa Barbara County Fire Department, county parcel and structure boundaries, and satellite imagery. Additional variables calculated include Normalized Difference Vegetation Index (NDVI; a measure of greenness), regional precipitation, and landcover type. Using these variables, a dataset was compiled for each structure that had been inspected by SBCFD and used to train a machine learning model.

In the Model Creation phase, a Random Forest Model was then initiated and run on the training dataset, generating a binary compliance prediction. We then tested the model for accuracy based on known compliant or non-compliant properties, and corresponding changes to parameters and predictors were made and re-tested as necessary.

The Random Forest model is ideal for this analysis as new variables can easily be added, and the model generates a score for each of the variables' importance to the model's results. These scores show which variables have the greatest influence on the model's predictions, guiding us toward the most effective ways to refine and improve the model's performance.

The Reproducible Pipeline phase involved carefully documenting the project workflow for future researchers. A user guide for how to use the model to generate new predictions on unseen data was also written. Data visualizations were designed to display predictions from the model about which properties are presently in compliance.

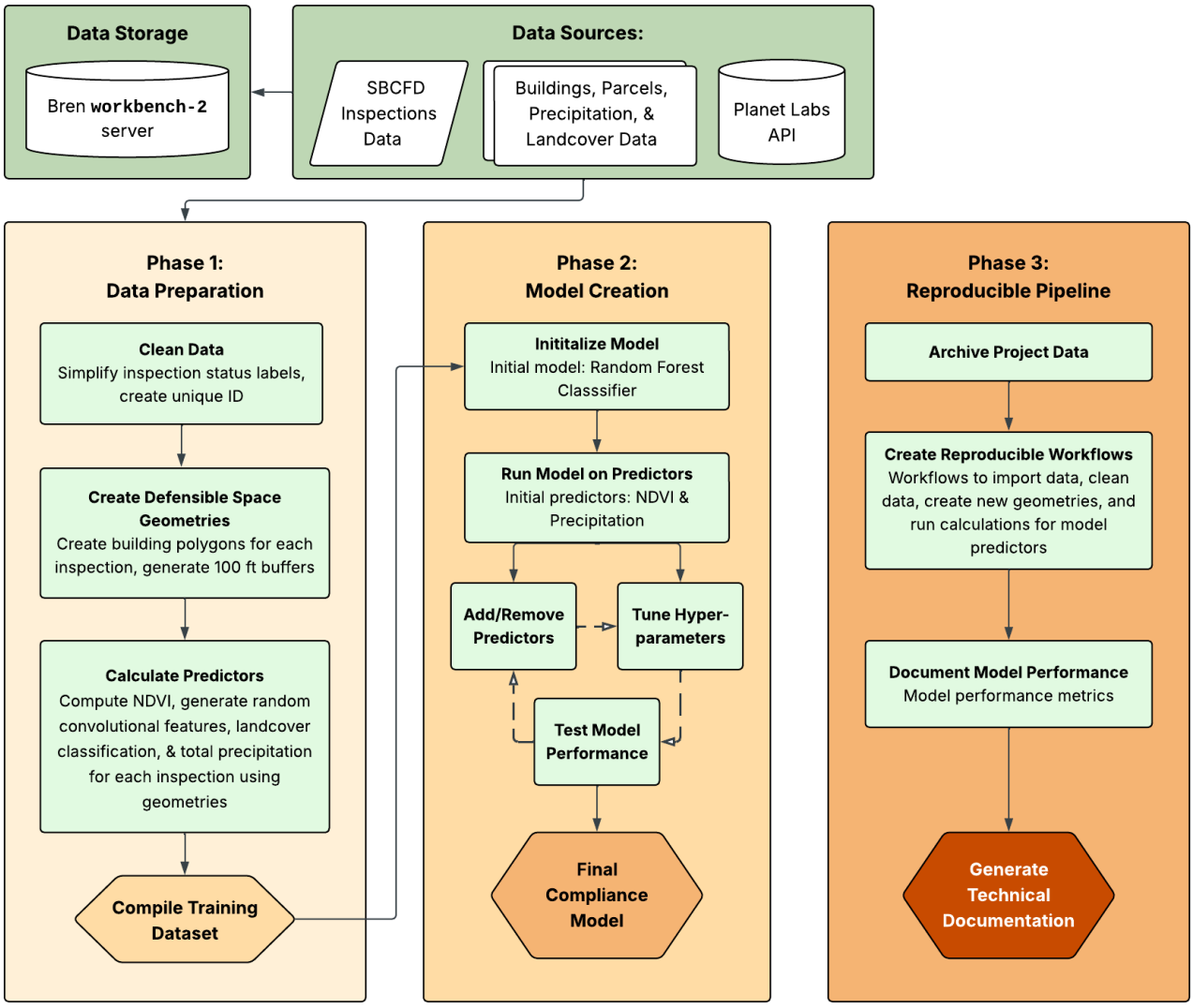

Figure 2: Project Approach Flowchart, depicting data sources, storage, & preparation; model creation; data visualization; and reproducible pipeline process.

II. METHODS

2.1 Data Sources

Please see the following for a table of all data (with sources) used in this project.

Table 1: Data Summary

Dataset	Type, Approx. Size	Source & Citation	Use
SBCFD Defensible Space Inspections (2019 - 2023)	.shp, .gdb. (1.89 GB)	Provided by SBCFD	Model training data (Compliance status labels, inspection points)
Defensible Space Data Dictionary	.xlsx (102 KB)	Provided by SBCFD	Metadata, variable definitions
Planet Labs Satellite Imagery (Basemaps) & Metadata	.tif, .json (111 GB)	Planet Labs PBC. (2019-2023). *Planet application program interface: In space for life on Earth*. Planet. https://api.planet.com	Model training data (random convolutional features)
Planet Labs Satellite Imagery (Analytic Surface Reflectance) & Metadata	.tif, .json (760 GB)	Planet Labs PBC. (2019-2023). *Planet application program interface: In space for life on Earth*. Planet. https://api.planet.com	Model training data (average NDVI)
Sentinel-2 L2A Multispectral Imagery	NA	Microsoft Planetary Computer (2018-2023). *Sentinel 2 Level 2A*. Microsoft. https://planetarycomputer.microsoft.com/api/stac/v1/collections/sentinel-2-l2a	Model training data (average NDVI)
Parcels	.gdb (268 MB)	Santa Barbara County Assessor. (2020). *Santa Barbara County Assessor Tax Parcels Layer.* *https://databasin.org/datasets/008da0ba3dc14be18aa3bcfd90dc9615/*	Model training data
California Building Footprints	.geojson (3.5GB)	Bing Maps Team. (2018). *Computer generated building footprints for the United States*.	Model training data

		Microsoft	
Santa Barbara County Boundary	.shp (3.5 MB)	U.S. Census Bureau. (2020). TIGER/Line geodatabases.	Model training data
Rainfall data	.xslx (2.2 GB)	Santa Barbara County Hydrology Section	Model training data (average rainfall)
LANDFIRE Terrestrial Ecosystems Data	.csv (40 KB)	U.S. Geological Survey Gap Analysis Project. (2016). *GAP/LANDFIRE National Terrestrial Ecosystems 2011*. U.S. Geological Survey. https://doi.org/10.5066/F7ZS2TM0	Model training data (landcover type)

SBCFD Defensible Space Inspections

This project relies on defensible space compliance inspections data produced by SBCFD, for years 2019 to 2023 ("Defensible Space Inspections 2019-2023", referred to as "inspections"). Along with the inspections datasets for each year, our clients provided a data dictionary ("Defensible Space Data Dictionary") containing definitions and clarifications associated with variables from the inspections data. These data contain compliance status, structure type, year and day of inspection, and more data collected during the fire department's yearly inspections.

Figure 3: Location of non-compliant inspections (red points) superimposed on a map of Santa Barbara County, for years 2019-2023.

Planetscope Orthorectified Surface Reflectance Imagery (PSScene ortho_analytic_4b_sr)
Our imagery ("Planet Labs Satellite Imagery") was obtained using the Planet Labs Orders API.

Communication with the Planet API requires an HTTP authentication and API key, the former accomplished with *HTTPBasicAuth()* from python's *requests* package, and the ladder being found on the dashboard page for one's Planet Labs account. Planet has several types of APIs, two of which were used in this step (Data API and Orders API). They can be accessed through the URLs "https://api.planet.com/data/v1" and "https://api.planet.com/compute/ops/orders/v2" respectively. Pinging either API should return a "*200*" response, if communicating with the API correctly.

Planet APIs accept filters to specify which scenes to return upon a request to the Data API. We provided the following filters: geometry, asset, date range, and cloud cover range. The geometry filter selects for scenes that intersect with a specified AOI. We used an AOI consisting of approximately 60 polygons and over 40 vertices that encompass nearly every parcel. This can be found in *code/utils/geojson_io.geojson*, within the data-preparation repository.

The asset filter selects for imagery that contains the asset type "*ortho_analytic_4b*", where *ortho_analytic* refers to analysis ready orthorectified imagery and *4b* refers to 4 band imagery. This filter is necessary because otherwise the API will attempt to return orthorectified 4 band imagery for scenes where such imagery does not exist, causing an error. The date range and cloud cover filters simply selects for imagery within specified ranges of each, with the cloud cover imagery selecting for <1% cloud cover, and the date range filter iterating through each month in each year.

The filters were combined with a specification for the product bundle of interest (*PSScene*/*PlanetScope*) and made a request to the data API to fetch all scene IDs that fall within the filter specifications in a list format. These scenes are then taken and funneled into the Orders API for ordering.

The Orders API also supports image processing at the point of download through its tools feature. The imagery was then masked using the clip tool by specified AOIs, and the reproject tool to reproject the images to EPSG:3310/NAD83/California Albers.

Orders that have finished successfully can then be downloaded in a zip format. Zip files can be downloaded by pinging the Orders API for order IDs, then using the Planet library to request a download to a given local filepath, for each order.

This workflow also includes error handling and retry logic, as it will frequently stall and error without such. Errors of note include JSONDecodeError and ChunkedEncodingError. Such errors are often a result of connection slowdowns and interruptions with the API. While the code more often than not handles these errors to prevent interruption of the workflow, it's not fully resilient to stopping due to these errors and may still require some degree of supervision.

Future Research

API errors can be handled with an exponential backoff in a try/except block, in which the workflow waits an amount of time before retrying a ping that increases exponentially after each failure. Handing the API thousands of small polygons rather than several larger ones is likely bad practice, as it resulted in Planet returning incomplete data.

2.2 PHASE I: Data Preparation and Training Set Creation

Cleaning Inspections Data

The "status" column for each year's inspection data was simplified to either "Compliant", "Non-Compliant" or "Other", and then observations labeled "Other" were removed. Each inspection was then assigned an unique identifier.

Defensible Space Training Geometries

Cleaned inspection coordinates were spatially joined to the Microsoft Building Footprint Dataset. For inspections without buildings, rectangular polygons were created with an area equal to the average area of all inspections with buildings. These were then assigned to the inspection points missing building geometries.

Hundred-foot buffers were created around each building to represent the Zone 1 and Zone 2 defensible space buffer inspected by the fire department for compliance. To accurately establish property owner responsibility, each buffer was clipped to County Assessor parcel boundaries. For buildings overlapping parcel boundaries, they were assigned to the parcel that contained the greatest proportion of building area. This ensures that each clipped training geometry represents the area inspected by the SBCFD as closely as possible.

Image Featurization

To investigate whether or not satellite imagery alone could be used to predict compliance, visual spectrum imagery was downloaded. For this method, unclipped training geometries were used that had the potential to overlap neighboring properties. This is because the image featurization algorithm requires standardized shapes of geometries, and the process of clipping each buffer by parcel geometry results in more irregular polygons.

For each observation, the satellite image was clipped from the month prior for each training geometry. Because Planet Basemaps are monthly aggregates, and inspections occur at irregular times throughout the month, picking the month prior ensures that the satellite image would correspond to the state of the property's vegetation in the period preceding each inspection.

Using these clipped images, a featurization function borrowed from the MOSAIKS process was used to construct random convolution features (RCFs). RCFs capture a flexible measure of similarity between every sub-image across every pair of images into a K-dimensional feature

vector.[1] This process converts attributes from visual imagery into tabular data format. The features are joined back to the inspections using their unique identifier.

Microsoft Planetary Computer Sentinel-2 Level 2A Imagery

The satellite imagery used to calculate NDVI for each property was sourced using Microsoft Planetary Computer's data catalog. Level 2A Data from the Sentinel-2 satellite was chosen for its global multispectral (13-band) imagery, its resolution, and temporal coverage. The red and near-infrared (NIR) bands necessary for calculating NDVI both have 10m resolution and global coverage every 10 days.

Tiles of multispectral Sentinel-2 Level 2A data was sourced using Python from the Planetary Computer STAC API. A function was constructed to iterate through the inspections dataset, find imagery tiles that intersected each individual inspection geometry, crop to that geometry, and calculate the mean NDVI across the cropped image. This function uses the inspection training geometries, mentioned earlier.

The function automatically collects the values from the 'year' and 'month' columns of each inspection to query the Planetary Computer catalog. After sourcing a multispectral image with <25% cloud cover and cropping it to the inspection geometry, it then pulls the red and near-infrared bands from the image and calculates an average NDVI value across that image. The value is then appended to the original inspections dataset, and the function continues to the next inspection. Each queried image is approximately 30x30m, significantly cutting down on processing time.

This function is also built to input 'NaN' values to any row that encounters an error, such as RasterIOError (when the *rasterio* package fails to read a raster image), or APIError (when the API times out). In all, the function inputs less than 100 'NaN' values as a result of these errors, out of the 67,580 inspections in the dataset.

Code is also included for this function to subset large datasets into more manageable sections, writing .csv files for each, containing only the inspection identifier and the mean NDVI value. These files can then be concatenated into a single large .csv file using the notebook *ndvi_concat.ipynb* in the *Data-Preparation* repository. This is done to avoid runtime timeouts, as the functions can take time to iterate through and calculate NDVI for every inspection.

These functions are reproducible so long as each inspection in the dataset has an associated geographic point. They can be found in the Data-Preparation Repository under the folder *07_ndvi_pystac* in the *Data-Preparation* repository.

[1] Rolf, E., Proctor, J., Carleton, T. *et al.* A generalizable and accessible approach to machine learning with global satellite imagery. *Nat Commun* 12, 4392 (2021). https://doi.org/10.1038/s41467-021-24638-z

Training Set Creation

For our training set, the geometries clipped to the parcel boundaries were used to calculate spatially-derived attributes for our rainfall and landcover predictors. For NDVI, mean NDVI for each training geometry was calculated. For rainfall, precipitation data was obtained for the nearest rain gauge. For land cover classification, the number of pixels were counted from the landcover raster data that fell within each parcel boundary, and the majority landcover class was identified.

Each of these predictors were compiled into a tabular dataset, and joined with the unique identifier column from our inspections data.

2.3 PHASE II: Defensible Space Compliance Model

Model Selection

Our initial model was a Random Forest Classifier model. While many types of models must sacrifice predictive power for flexibility or vice versa, random forest models strike a strong balance between both. Its structure also makes it easier to add data parameters in the future. An 80-20 training-to-test split was used on the data for modeling.

Model Tuning

To improve model accuracy, model hyperparameters were tuned using cross validation, with the functions *GridSearchCV()* and *HalvingSearchCV().* The cross validation searches test many different combinations of hyperparameters on the training and test data, and selects which combinations result in the best model performance.

III. RESULTS REPORT

3.1 Training Dataset Creation

The training dataset was built from several different .csv files that contained the unique inspection identifier and the calculated results from each predictor. For example, the landcover classification .csv file contains only two columns: the unique identifier and the landcover classification code. These .csv files were then joined using the unique identifier, where every row represented a distinct inspection and the calculated predictors.

3.2 Initial Compliance Machine Learning Models

Initial trials of the Random Forest Classifier model were run on a downsampled subset of the training data to address the class imbalance, increasing the proportion of non-compliant observations to 25%. The idea behind this was to increase the ratio of non-compliant properties to compliant, as the original dataset contained >99% compliant properties. This resulted in an accuracy score of 74% (defining overall prediction correctness), and precision score of 40% (defining correctness of non-compliant predictions). Precision assesses the model's ability to correctly identify non-compliant properties and reduce false-positives, and is therefore the

most important metric for our model. Precision can be improved by tuning model hyperparameters and/or rebalancing the input data.

3.3 Final Compliance Machine Learning Models

3.3.1 Data Sampling

The initial Random Forest Classifier model was fit to a downsampled subset, removing majority class (compliant) rows from the data until minority class (non-compliant) rows made up 25% of the training data. Because non-compliant rows made up <1% of the dataset, this significantly reduced the amount of training data for the model and produced some of the weakest results.

Final models used an approach where the majority class was downsampled by 10%, and the minority class was upsampled to be 5% of the number of rows from that downsampled majority class. This increased the number of correct minority class predictions, although misclassified more than half of the majority class. In testing different quantities of upsampled/downsampled rows, no combination was found that eliminated false positives.

3.3.2 Random Forest Classifier Model

A Random Forest Classifier model was fit on an upsampled dataset, as opposed to the initial downsampling approach. This dataset upsampled the number of non-compliant properties in the training data to be 25% of the total, instead of the initial 0.06%. Fitting the model, the Random Forest was able to classify compliant inspections well, but failed to classify any inspections as non-compliant.

To refine this model, the *sklearn* HalvingRandomSearchCV method was used to identify the best hyperparameters. This method works by fitting the model on smaller subsets of the data, finding the most promising hyperparameters quickly, and dropping the least promising ones. Through multiple iterations of training using different combinations of hyperparameters, only the best combinations remain and the highest performing combination is selected. The model is then fit a final time on that best combination.

After using HalvingRandomSearchCV, this model showed little to no improvement, and still failed to classify any inspections as non-compliant.

3.3.3 XGBoost Classifier Model

Another strong choice for this dataset was the XGBoost Classifier, as it can handle NA values and mixed-type features. This model works by creating many simple decision trees and recursively checks the errors of the previous tree, attempting to correct those errors with each tree. This model was fit to an identical upsampled dataset as the Random Forest Classifier.

After refining with HalvingRandomSearchCV, this model produced nearly exactly the same result as the Random Forest Classifier, classifying compliant inspections well, but failing to classify any inspections as non-compliant.

3.3.4 Machine Learning Models Conclusion

Both the Random Forest and XGBoost Classifiers failed to classify any non-compliant inspections correctly, with a default threshold of confidence. Adjusting this threshold to 90% confidence allowed us to correctly classify some non-compliant properties, but also misclassifying a greater number of compliant properties as non-compliant. When visualizing model results using a ROC AUC (area under curve), both models did not achieve higher than an AUC score of 0.70, where random guessing has an AUC score of 0.50.

It is possible that the models could be improved with higher resolution satellite imagery of the parcels, which could give more specific image classification values. Another possibility is that using more predictor variables specific to each parcel (such as distance to wildland interface zones, or topography) could improve the results. The performance of the model is also limited by the lack of data for non-compliant inspections.

3.4 Reproducibility

The Python notebooks for the Random Forest Classifier and XGBoost models are both available in the Modeling Github repository. Both are built to accept a new data file, and the desired columns to input into the model can be selected in the following code chunk. Both notebooks include code to upsample the minority class, downsample the majority class, or both at once. Running the entire notebook on a given dataset will print a ROC AUC Curve and a Confusion Matrix for the predictions.

3.2 Limitations

The compliance dataset contained a significant class imbalance problem, with compliant properties making up the majority of the data at over 99%. The lack of data for non-compliant properties severely impeded the model's training data, causing it to be overfamiliar with compliant parcels. However, this model could be more effective in other counties or regions that have higher rates of non-compliance with defensible space.

The geographic specificity of this model creates inherent generalizability constraints. The trained model reflects local patterns of compliance influenced by the accuracy of the fire department's classifications. SBCFD conducts these inspections via truck, with inspectors often examining the property from the truck or public street, which limits their ability to assess brush clearance as the majority of a property may not be visible. While our model incorporates numerous features that we believe may be characteristic of a compliant property, the uncertainty generated during the inspections process limits this model's generalizability and

accuracy. We were unable to incorporate other qualitative compliance determinants (such as property owner attitudes) that could be significant predictors of compliance.

IV. PRODUCT DESCRIPTION

4.1 Data Management Pipeline to Compile Training Data

After receiving inspections data from 2018-2023 from SBCFD, a Python script was written to clean and concatenate them into a single dataframe. A unique identifying number (inspection ID) was then given to each individual inspection.

Using the locations, parcel geometries, and satellite imagery of each property, variables for landcover type, local precipitation total, and NDVI were calculated in Python. These variables were then written to individual .csv files, containing only the inspection ID and the calculated variable. These .csv files were then merged by the unique identifier into the training dataset, with each column representing a calculated variable for the given inspection ID. Section 4.3 - Updating Models and Outputs will describe which specific notebooks should be run to replicate this process.

4.2 Modeling Code

The model used in this project is a Random Forest Classifier model, using the *sklearn* package in Python. The model is designed to split the dataset into two subsets: a training subset containing 70% of the data, and a testing subset containing 30%. At this point, initial model predictions are made. The model can then be refined by adjusting the model's hyperparameters, such as the number of decision trees, the maximum depth of each tree, and the number of samples required to split a node. Hyperparameter tuning for our model was done using a randomized search and cross validation, to test different hyperparameter combinations and find the combination with the best predictive performance. Once the best hyperparameter combination is incorporated into the model, it is re-trained and evaluated using metrics like Accuracy, Precision, Recall, Receiver Operating Characteristic (ROC), and Area Under Curve (AUC). Finally, the importance scores of each variable in the dataset are examined to see which variables most influence predictions.

4.3 Workflows to Update Models and Outputs

Adding Additional Inspections Data

To add additional years of inspections data to the model, inspections should first be cleaned and processed following the notebooks in the data preparation repository. The "status" column that indicates compliance status should simplify observations to either "Compliant" or "Non-Compliant", with inspections not falling under either category removed. Then, a unique identifier should be added, with the ID sequentially following the last number in the

"inspection_id" column. This process occurs in the "data-preparation/code/00_label_data/00_clean_inspections.ipynb" notebook.

Next, buildings data should be joined to inspections using the workflow outlined in the "data-preparation/code/00_label_data/03_building_polygons.ipynb" notebook. To calculate buffer geometries around each building, follow the workflow in the "data-preparation/code/00_label_data/04_buffer_geometry.ipynb" notebook. Once buffer geometries are calculated for each inspection, concatenate the new year of inspections data to the prior years using the workflow in "data-preparation/code/00_label_data/05_concatenate_inspections.ipynb".

Computing Features on New Inspections Data

To compute random convolutional features on new years of inspections data, first download new years of Planet Basemaps imagery using the workflow detailed in "data-preparation/code/01_satellite_imagery/00_download_basemaps.ipynb". Next, clip downloaded imagery using the process detailed in "data-preparation/code/01_satellite_imagery/03_clip_basemap_images.ipynb". Then, compute MOSAIKS features on these clipped images using "data-preparation/code/06_model_development/01_mosaiks.py".

Computing NDVI on New Inspections Data

To compute NDVI on additional years of inspections data, follow the workflow in "data-preparation/code/07_ndvi_pystac/ndvi_calculate.ipynb" to access Sentinel 2 imagery and calculate NDVI.

Updating Models with New Inspections Data

To update models with additional years of inspections data, incorporate MOSAIKS features, and include new NDVI calculations, follow the workflow in "modeling/code/00_random_forest/04_random_forest.ipynb".

VI. ARCHIVE ACCESS

6.1 Shared data folder

Data for this project was hosted on the Bren School's "workbench-2" server. Non-imagery data (inspections, building data, etc) was stored under "/capstone/wildfire_prep/data", while imagery data was stored under "/data/wildfire_prep". Below are the file trees for each of these directories.

```
.
├── buffer_geometries
├── ca_counties
│   └── tl_2023_06083_roads
├── features
```

```
│   └── features_mosaiks_4000.feather
├── inspections_data
│   ├── 2018_defensible_space_inspections_SBC
│   ├── 2019_defensible_space_inspections_SBC
│   ├── 2020_defensible_space_inspections_SBC
│   ├── 2021_defensible_space_inspections_SBC
│   ├── 2022_defensible_space_inspections_SBC
│   ├── 2023_defensible_space_inspections_SBC.gdb
│   ├── cleaned_status
│   ├── correspondence with SB Fire re data.pdf
│   └── Defensible Space Data Dictionary (variables)_Marsh edit 2.0.xlsx
├── joined_inspections_buildings
├── joined_inspections_parcels
├── metadata
│   ├── landcover_codes.csv
│   └── structuretype_codes.csv
├── microsoft_buildings
├── parcel_boundaries
├── parcel_data
│   └── Defensible_Space_2024_address_points.gdb
├── Parcel_Maps_SpatialIndex_2024
├── PUZZLE_PIECES
│   ├── assemble_puzzle.ipynb
│   ├── assembled_puzzle.csv
│   ├── inspection_id_features.csv
│   ├── inspection_id_landcover_buffer_geometries.csv
│   ├── inspection_id_landcover_training_geometries.csv
│   ├── inspections_id_compliance_status.csv
│   ├── inspections_id_structure_type.csv
│   ├── inspections_master_training_geometries.geojson
│   └── inspections_master.geojson
├── sb_rain_gauges
├── sb_roads_2023
└── training_geometries
```

Figure 4: ASCII Tree file structure for this project's main data folder, located on UCSB Bren's `workbench-2` server at `/capstone/wildfire_prep/data`

```
.
├── basemaps
├── clipped_basemaps
└── full_planet
```

Figure 5: ASCII Tree file structure for this project's imagery data folder, located on UCSB Bren's `workbench-2` server at `/data/wildfire_prep`

6.2 Github Organization: WildfirePrep

Information about this project, as well as all the following repositories are housed in the "wildfire-prep" GitHub organization, located at https://github.com/wildfire-prep. Additional information, including project summary, data sources, and authors & contributors can be found in the organization's README.md.

6.3 Github Repository: Data Preparation

This repository, located at https://github.com/wildfire-prep/data-preparation contains code and workflows used in Phase I of our project. This repository is split into directories for labeling and cleaning data, retrieving satellite imagery, clipping and calculating variables from satellite imagery, processing rainfall & landcover data, and feature extraction. Additionally, it contains utility and configuration files for storing commonly used variables, custom functions, and environment setup.

6.3 Github Repository: Modeling

This repository, located at https://github.com/wildfire-prep/modeling contains code for different modeling attempts used in Phase II of our project. This repository is split into directories for the different types of models tested, and different combinations of predictors incorporated.

VII. REFERENCES

1. U.S. Department of Homeland Security. (2020, March). *Home Builder's Guide to Construction in Wildfire Zones - Defensible Space*. Ready.gov. https://www.ready.gov/sites/default/files/2020-03/home-builder-guide-construction-defensible-space.pdf.

2. Rolf, E., Proctor, J., Carleton, T. *et al.* A generalizable and accessible approach to machine learning with global satellite imagery. *Nat Commun* 12, 4392 (2021). https://doi.org/10.1038/s41467-021-24638-z

3. CalFire. (2024, Jan). *Defensible Space and the Law: CAL FIRE's Guide to Defensible Space Requirements.* Readyforwildfire.org. https://readyforwildfire.org/wp-content/uploads/2024/07/Defensible-Space-and-the-Law-Factsheet-Revisions-Jan-2024.pdf

4. Office of the State Fire Marshal. (2025). *Fire hazard severity zones*. California Department of Forestry and Fire Protection. https://osfm.fire.ca.gov/what-we-do/community-wildfire-preparedness-and-mitigation/fire-hazard-severity-zones

5. Santa Barbara County Fire Safe Council. (n.d.). *How to Prepare: Defensible space*. https://sbfiresafecouncil.org/defensible-space/

6. U.S. Geological Survey Gap Analysis Project. (2016). *GAP/LANDFIRE National Terrestrial Ecosystems 2011.* U.S. Geological Survey. https://doi.org/10.5066/F7ZS2TM0