

**Leveraging Automation in Data Enhancement and Quality Control Protocols  
for Post-Windstorm Reconnaissance Data**

by

Hadiyah Rawajfih

A thesis submitted to the Graduate Faculty of  
Auburn University  
in partial fulfillment of the  
requirements for the Degree of  
Master of Science

Auburn, Alabama

May 1, 2021

Keywords: reconnaissance, automation, machine learning

Copyright 2021 by Hadiyah Rawajfih

Approved by

Dr. David Roueche, Chair, Assistant Professor of Civil and Environmental Engineering

Dr. Justin D. Marshall, Associate Professor of Civil and Environmental Engineering

Dr. Jeffrey LaMondia, Associate Professor of Civil and Environmental Engineering

## Abstract

Extreme windstorms cause loss of life, major financial losses, and disrupt community well-being. Post-windstorm field reconnaissance is one of the traditional methods used to collect data about these natural hazards. The analysis of this data helps us better understand these disasters and improve the performance of buildings and other infrastructure during such extreme events. The raw data collected from reconnaissance missions is often fragmented and nonuniform, which necessitates formalizing data enhancement and quality control protocols that increase the completeness and accuracy of these datasets. This thesis presents data enhancement and quality control protocols for ensuring that the building performance datasets are accurate, complete, and standardized, making them suitable for analysis and further use by the natural hazards engineering community. However, this data enhancement and quality control process can take months to complete, delaying data analysis. This thesis also demonstrates a preliminary framework for automating key components of the data enhancement and quality control process to reduce the time required. The automation framework uses modern technologies that include web scraping and the use of established machine learning models from past works to classify damage based on imagery. The results of comparing the human approach for the data enhancement and quality control process to the automation framework shows promise in incorporating automation in post-windstorm field reconnaissance. While the accuracy of the framework is not as high as the human approach, continuous improvements can be made to the individual components of the framework to increase the accuracy. Ultimately, this approach aims to ensure high quality, standardized post-windstorm reconnaissance datasets can be generated and published rapidly after extreme windstorms and used to strengthen the resilience of at-risk communities.

## Table of Contents

Abstract . . . . .	ii
List of Figures . . . . .	vi
List of Tables . . . . .	ix
1 Introduction . . . . .	1
1.1 Objectives . . . . .	3
1.2 Thesis Outline . . . . .	4
2 Literature Review . . . . .	5
2.1 Overview of Post-windstorm Reconnaissance Developments . . . . .	5
2.1.1 The Importance of Post-windstorm Reconnaissance Data . . . . .	5
2.1.2 Organizational Contributors to the Collection of Post-windstorm Reconnaissance Data . . . . .	7
2.1.3 Natural Hazards Research Community Efforts . . . . .	9
2.2 Post-windstorm Reconnaissance in Digital Format . . . . .	13
2.3 NSF Structural Extreme Event Reconnaissance (StEER) . . . . .	14
2.4 Automating Damage Detection and Classification Using Machine Learning . . . . .	16
2.4.1 Notable Work on Automating Damage detection . . . . .	17
2.4.2 Machine Learning and Deep Learning Technologies in Damage Classification . . . . .	17
2.4.3 Automating the Detection of Building Attributes . . . . .	18
2.5 This Thesis . . . . .	19
2.6 Chapter Summary . . . . .	20
3 Data Enhancement and Quality Control Protocol . . . . .	21
3.1 Framework for Data Enhancement and Quality Control . . . . .	22

3.2	Pre-processing Tasks and Considerations . . . . .	23
3.3	Main Processing Tasks and Considerations . . . . .	25
3.3.1	Extension of DE/QC to Virtual Assessments . . . . .	29
3.3.2	StEER Windstorm Building Assessment Survey Instrument . . . . .	30
3.3.3	Supplemental Data Sources . . . . .	34
3.3.4	Logistics of the DE/QC Process . . . . .	38
3.4	Post Processing Tasks and Considerations . . . . .	43
3.4.1	Accuracy . . . . .	44
3.4.2	Currentness and Completeness . . . . .	47
3.4.3	Consistency . . . . .	48
3.5	Data Librarians . . . . .	48
3.6	Case Study . . . . .	50
3.7	StEER Data Archive . . . . .	55
3.8	Chapter Summary . . . . .	57
4	Automation Framework . . . . .	59
4.1	Framework . . . . .	59
4.2	Automation Techniques and Tools . . . . .	62
4.2.1	Web scraping Process . . . . .	62
4.2.2	BRAILS Overview and Incorporation in the Automation Framework . . . . .	63
4.2.3	Data Enhancement Using SURF . . . . .	65
4.2.4	NOAA Aerial Imagery in the Automation Framework . . . . .	66
4.3	Classifying Roof Damage Using Machine Learning Algorithms . . . . .	67
4.3.1	Exploring Different Algorithms and Approaches . . . . .	72
4.3.2	Stochastic Gradient Descent (SGD) Classifier . . . . .	73
4.3.3	Support Vector Machines (SVC) Classifier . . . . .	77
4.3.4	Multi-layer Perceptron (MLP) Classifier . . . . .	80
4.3.5	Testing Tarped versus Untarped Roofs . . . . .	82

4.3.6	Final Model Selection . . . . .	83
4.4	Framework Performance Evaluation and Comparison to Human Collected Data	84
4.4.1	Web Scraping Vs. BRAILS AI Approach Vs. StEER Collected Data	85
4.4.2	Comparing SURF Enhanced Web Scraping Results with StEER's Data	92
4.4.3	Comparing Damage Predictions to StEER's Data . . . . .	92
4.5	Future Work and Improvement . . . . .	97
4.6	Chapter Summary . . . . .	99
5	Summary and Conclusion . . . . .	100
	Appendices . . . . .	120
A	StEER Buildings - Windstorm Application Fields . . . . .	121
B	Automation Framework Code . . . . .	145

## List of Figures

1.1	Hurricane and tornado tracks that occurred between 2017-2020 . . . . .	2
2.1	CONVERGE’s multidisciplinary networks . . . . .	9
3.1	Distribution of StEER’s records per windstorm that was surveyed by FAST . . .	22
3.2	Flowchart of the framework for data enhancement and quality control . . . . .	23
3.3	Location validation for Hurricane Michael Dataset . . . . .	24
3.4	Table view in StEER’s archive windstorm Fulcrum application . . . . .	31
3.5	Nashville Tornado (2020) example record in Fulcrum . . . . .	32
3.6	StEER’s Wind Damage Rating criteria which can be found in the DE/QC handbook	33
3.7	Comparison between NOAA aerial imagery and D2D assessment for the same structure. . . . .	36
3.8	Point Clouds captured by the FAST in field reconnaissance . . . . .	37
3.9	Pictometry imagery of a coastal residential structure impacted by Hurricane Michael in 2018 . . . . .	37
3.10	Data sources available for a record from the Nashville Tornado (2020) . . . . .	39
3.11	The Fulcrum platform landing page . . . . .	40
3.12	The Fulcrum mobile application interface . . . . .	41

3.13	Map view in StEER’s archive windstorm Fulcrum application . . . . .	42
3.14	Map and table view in StEER’s archive windstorm Fulcrum application . . . . .	43
3.15	Example of Slack workspace communications . . . . .	51
3.16	Example one of how to anchor the different sides of the structure based on details around the structure . . . . .	53
3.17	Example two of how to anchor the different sides of the structure based on details around the structure . . . . .	53
3.18	Benefit of using street-level panoramic imagery in addition to D2D photographs	54
3.19	StEER’s data distribution into QC codes . . . . .	57
4.1	Automation framework flowchart . . . . .	60
4.2	scikit-learn library flowchart that gives guidance on how to choose the appropriate machine learning algorithm . . . . .	70
4.3	Confusion matrix for SGDClassifier results . . . . .	76
4.4	Confusion matrix for SVC results . . . . .	80
4.5	Confusion matrix for MLPClassifier results . . . . .	84
4.6	Confusion matrix showing the performance of BRAILS occupancy classifier rela- tive to StEER’s data . . . . .	88
4.7	Examples of structures that have complex roofs based on StEER’s DE/QC hand- book guidelines . . . . .	89
4.8	Two images of the same structure to compare results from web scraping, BRAILS, and StEER’s data . . . . .	91

4.9	Two images of the same structure to compare results from web scraping, BRAILS, and StEER’s data . . . . .	91
4.10	Pie chart that shows the difference between the true value and predicted value for the images classified using NOAA aerial imagery with zoom level 20 . . . . .	95
4.11	Roofs that were classified as undamaged but based on StEER’s data are labeled as minor damage . . . . .	95
4.12	Roofs that were classified as minor but based on StEER’s data are labeled as undamaged. These images show how easy it is to misclassify minor and undamaged roofs using aerial imagery only. . . . .	96
4.13	An example of a misclassified image. a) NOAA aerial imagery of the structure , b) An image of the same structure captured by the FAST . . . . .	96
4.14	Pie chart that shows the difference between the true value and predicted value for the images classified using NOAA aerial imagery with zoom level 18 . . . . .	97



## List of Tables

3.1	The major data classes in StEER’s datasets. . . . .	22
3.2	Examples of data found in county records before cleanup process . . . . .	26
3.3	Data extracted from county data after cleanup . . . . .	26
3.4	StEER’s QC codes, which are used to track the data enhancement and quality control progress . . . . .	28
3.5	Main data types available to use in data enhancement along with their sources.	35
3.6	Examples of fields that are multifiltered during quality control . . . . .	46
3.7	A list of all windstorm events that StEER responded to as well as the type of response provided . . . . .	55
4.1	Roof damage status criteria used in training the machine learning algorithms . .	69
4.2	The parameters tested for the SGDClassifier . . . . .	75
4.3	The best score parameters for the SGDClassifier . . . . .	76
4.4	The parameters tested for the SVC using GridSearchCV . . . . .	79
4.5	The best score parameters for the SVC . . . . .	79
4.6	The parameters tested for the MLPClassifier using GridSearchCV . . . . .	81
4.7	The best score parameters for the MLP . . . . .	81
4.8	Summary of the performance of the tested models to classify damage into five roof damage classes. . . . .	84
4.9	Error percentages of StEER’s datasets used to evaluate automation framework .	85
4.10	The number of results found per attribute using web scraping . . . . .	86
4.11	Comparison between human approach from StEER, web scraping, and BRAILS	89

4.12 StEER's Occupancy field options and their equivalent BRAILS classes for the purpose of comparison. . . . .	90
4.13 A comparison of web scraping results after enhancement with SURF with StEER's data . . . . .	93
A.1 StEER Buildings - Windstorm Application Fields . . . . .	121

## Chapter 1

### Introduction

Between the years 1900 and 2017, 197 hurricanes impacted the continental United States causing an estimated US \$2 trillion in normalized (2018) damage, amounting to nearly US\$17 billion annually (Weinkle et al., 2018). In 2020 alone, 22 weather and climate disasters occurred, costing the US \$95 billion (Adam, 2020). The 2020 hurricane season was the most active on record, both in terms of storms and landfalls, with 30 named storms and 12 landfalls in the continental US according to the National Oceanic and Atmospheric Administration (NOAA, 2020). The hurricane impacts in 2020 were coupled with several destructive tornadoes, including two tornadoes that struck Tennessee in March 2020, killing 25 persons. The frequency of extreme windstorm hazards along with the devastation they incur underscores the critical need for improved resiliency of at-risk communities exposed to these hazards. The impacts also highlight the need for the natural hazards engineering community to better understand these hazards and more effectively transfer that knowledge to at-risk communities to improve their resilience. Figure 1.1 shows a plot of windstorm tracks that occurred between 2017 and 2020 along with the Saffir-Simpson category (Taylor et al., 2010) associated with each hurricane and the Enhanced Fujita (EF) rating (Marshall et al., 2004) for each tornado, illustrating the frequency and spatial distribution of these windstorms in the southeast particularly.

The frequency and devastating impacts of extreme windstorms make it imperative that the natural hazards research community learns from these events, and uses that knowledge to reduce future impacts. One of the primary means of learning from windstorm disasters has traditionally been through post-windstorm field reconnaissance missions. Post-windstorm reconnaissance refers to investigative trips taken to areas affected by windstorms to assess

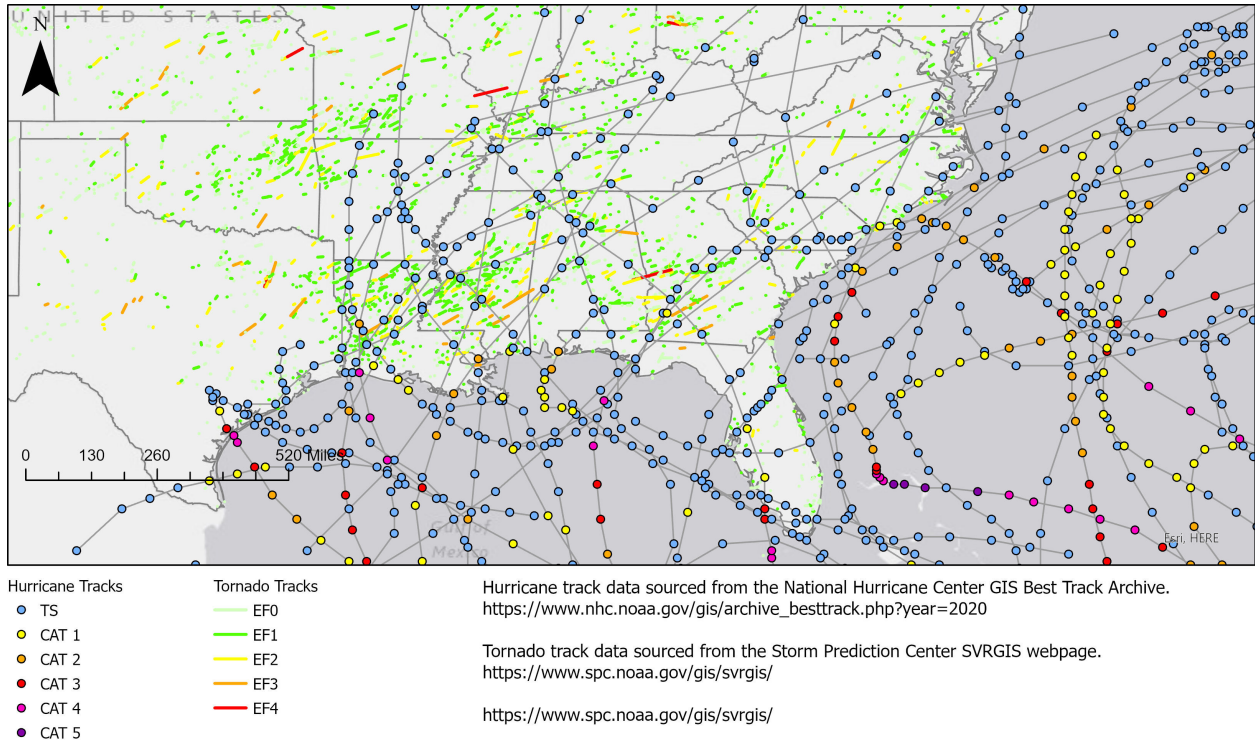


Figure 1.1: A plot of hurricane and tornado tracks that occurred between 2017-2020. The figure shows the category of each hurricane as well as the EF rating of each tornado. The hurricane track data was obtained from the National Hurricane Center GIS Best Track Archive. The tornado track data was sourced from the Storm Prediction Center SVRGIS webpage.

the performance of the built and natural environment. Such missions are typically conducted by engineers, scientists, and others in the natural hazards engineering community to collect perishable data. This data is perishable because it quickly gets altered from its original state due to clean up, repairs, and environmental elements. When conducting building damage assessments, in particular, investigative teams select areas of interest within the affected regions before the trip, and choose samples of structures affected within each of the selected areas. Collected data may include photographs, videos, notes, and measurements, which are subsequently used by the research community for further analyses. These analyses help identify the shortcomings in the design of buildings and the performance of construction methods and materials, generating knowledge that can be utilized to update building codes.

Previously, in post-disaster reconnaissance, there was often a trade-off between breadth and depth of the assessments, as the collected data would either represent a high quantity of assessments at superficial depth (e.g., broad canvassing using aerial imagery to assign overall damage ratings), or a small quantity of more detailed assessments (e.g., detailed building performance assessments on individual building basis). More recently, the incorporation of modern technologies has expanded the efficiency of the assessments, allowing for the collection of more detailed assessments over a larger number of structures. However, the limited time in the field available to collect the data often results in raw datasets that are fragmented and nonuniform, necessitating further postprocessing to enhance it. Postprocessing includes filling in the gaps and performing quality control. Thus, protocols to enhance the raw data and increase its accuracy are essential for accurate analysis.

However, postprocessing of reconnaissance data can take several months, due to the time required to organize the raw data files, identify and integrate supplemental data sources, manually review data and findings, etc. One way to increase efficiency is to automate as much of the postprocessing as possible, leveraging modern technologies such as machine learning and automation scripts. To develop and have confidence in such methods however, it is imperative to have high quality, labeled datasets to validate the automated methods against, and to have well-defined standards for the reconnaissance datasets that ensure the final, post-processed datasets meet the needs of the natural hazards research community for reuse and analysis.

## **1.1 Objectives**

The objectives of this thesis are to (1) present data enhancement and quality control protocol for producing robust, standardized post-windstorm building performance datasets that promote widespread reuse; and (2) demonstrate a preliminary framework for automating portions of the data enhancement and quality control process using machine learning, web scraping, and other big data techniques.

## 1.2 Thesis Outline

Chapter 2 is a literature review that follows the theme of the thesis objectives, firstly exploring past works related to post-windstorm reconnaissance and secondly summarizing past work related specifically to post-disaster damage detection using machine learning. Chapter 3 presents a data enhancement and quality control protocol for post-windstorm building performance datasets. Chapter 4 focuses on the components that go into the preliminary automation framework and discusses preliminary comparisons of the automated techniques against human-collected data. Finally, Chapter 5 presents a summary and conclusion of the work in this thesis.

## Chapter 2

### Literature Review

## 2.1 Overview of Post-windstorm Reconnaissance Developments

### 2.1.1 The Importance of Post-windstorm Reconnaissance Data

Natural hazards, such as windstorms and earthquakes, leave a tremendous amount of data in their wake. This includes event related metadata such as wind speeds and directions, windstorm path, etc., as well as data related to the impacts of the event to the built and natural environment. Such data is highly valuable but usually available for only a short period of time due to rescue operations, repair, and cleanup efforts. Therefore, timely reconnaissance missions soon after an event can save the data produced by the event for later processing. The analysis of reconnaissance data can then help us better understand the behavioral patterns and various impacts of natural hazards, which leads to new discoveries, building new data-driven models, as well as testing and verifying existing models (Wartman et al., 2020). However, the value of post-disaster reconnaissance extends beyond the documentation and understanding of natural hazards. Analyses performed on post-disaster damage assessments, which are an example of reconnaissance data, help enhance our understanding of the performance of the built environment under these extreme events, leading to the better understanding of structural shortcomings, assisting in developing better solutions to design more resilient structures, and aiding in finding mitigation measures.

In the United States, some of the earliest documented field reconnaissance missions were performed by Lawson (1908) and Baier (1897). Lawson (1908) collected data to document the 1906 M7.9 San Francisco earthquake and Reid (1910) developed the landmark theory of elastic rebound based on observations from field reconnaissance. Additionally, Baier (1897)

studied the impacts of the St. Louis Tornado that occurred in 1896 and was able to determine the pressures acting on buildings based on the wind speeds the buildings were subjected to. More recently, reconnaissance data of the built environment, including overall damage evaluations and component-level damage assessments, have been used to inform and validate damage prediction models for hurricanes at the regional-scale (Pinelli et al., 2004; Vickery et al., 2006), tornadoes at event scale (Jain et al., 2020), and at individual building scale (Li and Ellingwood, 2006). Building performance assessments collected in reconnaissance are also used in training and benchmarking automated techniques for damage recognition and classification (Pinelli et al., 2004; Sirmacek and Unsalan, 2009; Radhika et al., 2015; Thomas et al., 2014; Kashani et al., 2015).

Furthermore, reconnaissance data is essential in identifying critical weaknesses in building construction resilience. Kareem (1985) assessed the structural performance of buildings in Hurricane Alicia in 1983 and found that the poor performance of structures was primarily due to the deficiencies in hurricane-resistant construction in the impacted areas. Additionally, Keith and Rose (1994) investigated the performance of residential and low-rise commercial structures that were impacted by Hurricane Andrew in 1992. Moreover, van de Lindt et al. (2007) focused on surveying wood-frame structures and evaluating their performance in Hurricane Katrina which occurred in 2005. Based on observations from this survey, the authors were able to make recommendations for better construction and detailing for wood-frame structures. The authors also made recommendations to achieve better connections in wood systems. Many other examples can be found in Wartman et al. (2020) that emphasize the critical role field reconnaissance data plays in advancing hazards engineering and ultimately, community resilience.



### 2.1.2 Organizational Contributors to the Collection of Post-windstorm Reconnaissance Data

Today, various organizations contribute to the collection of post-windstorm reconnaissance data, including academic organizations, state and federal organizations, non-government organizations, and more. A major contributor in the US has been the Federal Emergency Management Agency (FEMA), which contributes to post-disaster reconnaissance data collection through the Mitigation Assessment Team Program (MAT). FEMA has been deploying MAT for over 30 years with a primary focus on evaluating the performance of buildings and infrastructure impacted by natural hazards. It also publishes recommendations for improving the performance of structures after assessing the damage from windstorm events as well as code improvement recommendations shown in (FEMA, 2014, 2018, 2019, 2020a). In addition to the reports and recovery advisories, FEMA also provides a variety of public information and data related to natural hazard events which can be found on the OpenFEMA repository, including disaster declaration summaries, datasets consisting of Community Emergency Response Teams and their information, flood financial impacts datasets, among other datasets (FEMA, 2020b).

The National Institute of Standards and Technology (NIST) also frequently contributes to post-windstorm reconnaissance data through the Disaster and Failure Studies program. It deploys teams to conduct building damage assessments due to natural hazards, assesses the performance of emergency communication systems present, and evaluates shelter and safe areas (Riley, 2002; Kuligowski et al., 2014; Main et al., 2021). Additionally, the program provides in-depth reports focusing on critical facilities such as Kuligowski et al. (2013), as well as case study reports such as Gross et al. (2010). NIST also outlined procedures to identify vulnerabilities of existing structures to seismic and windstorm hazards (Lew et al., 2002).

The National Science Foundation has recently made significant investments related to the collection and reuse of reconnaissance data, primarily through the Natural Hazards

Engineering Infrastructure (NHERI) program, which is a national organization that provides the natural hazards research community with infrastructure, education, and the necessary tools to help with natural hazards research operations (Blain et al., 2020). Within the broad umbrella of NHERI are experimental research facilities for wind, earthquakes, and tsunami hazards (Blain et al., 2020), a dedicated Natural Hazards Reconnaissance Facility (also known as the RAPID facility) that supplies hardware, software, and expertise to support reconnaissance missions, and the DesignSafe Cyberinfrastructure, which serves as a long-term data repository and analysis facility and includes dedicated tools and infrastructure to support reconnaissance missions (Rathje et al., 2020).

Another contributor is NHERI’s CONVERGE network, which is supported by NSF. CONVERGE encompasses several Extreme Events Reconnaissance and Research (EER) networks, which are also supported by NSF. Peek et al. (2020) outline the disciplines that these networks focus on, which include: geotechnical engineering, social sciences, structural engineering, nearshore systems, operations and systems engineering, sustainable material management, and interdisciplinary science and engineering. Figure 2.1, which is taken from (Peek et al., 2020), shows the EER networks that CONVERGE established up to this date. The Structural Extreme Events Reconnaissance Network (StEER) is a network of structural and other hazards engineering experts with the objective of capturing perishable data that will advance our understanding of structural performance and the role of structural engineering within community resilience. StEER is funded by NSF with a mandate to develop and maintain the infrastructure and training necessary to support rapid reconnaissance missions that collect high-quality data suitable for reuse by the broader natural hazards engineering community. It participates in field reconnaissance for various natural hazards including hurricanes, tornadoes, earthquakes, and tsunamis. The datasets resulting from these missions can be found on DesignSafe-CI. StEER responds to events using both virtual, on-site, and hybrid workflows. StEER also publishes event briefings and detailed reports in the form of Preliminary Virtual Reconnaissance Reports (PVRR) and Early Access Reconnaissance

Reports (EARR). These reports include analysis and observations from virtual and field assessments taking into account local building codes and construction practices of the affected areas, as well as recommended improvements for future events. An in-depth explanation of StEER’s operations and workflow can be found in (Kijewski-Correa et al., 2021).

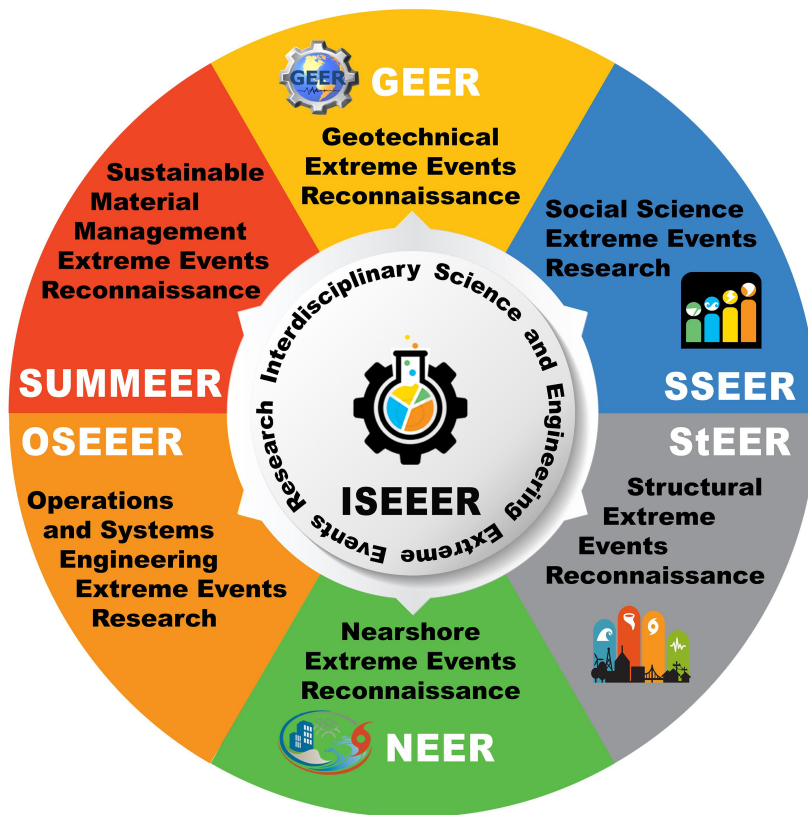


Figure 2.1: CONVERGE’s multidisciplinary networks, taken from Peek et al. (2020).

### 2.1.3 Natural Hazards Research Community Efforts

The interest in collecting and analyzing reconnaissance data is not limited to an organizational level. Throughout the years, the academic community has also studied the impacts of hazards through reconnaissance. The United States, in particular, has been hit with many major windstorms where many valuable lessons were learned from collected and processed reconnaissance data. Additionally, these efforts allowed for educated changes to building codes, as can be seen in Gurley and Masters (2011). The following is a discussion

on some of the major windstorms that hit the United States and the efforts made to study them.

One of the most catastrophic hurricanes that made landfall in the U.S was Hurricane Andrew in 1992. Keith and Rose (1994) investigated the performance of residential buildings in south Florida after Hurricane Andrew. The authors collected detailed damage observations of low-rise residential and commercial buildings of multiple types of construction including wood, steel, masonry, and concrete. From these observations it was concluded that the major cause of the poor performance of the investigated buildings was largely attributed to bad construction practices, especially in connections and detailing. Wakimoto and Black (1994) documented the destruction that occurred due to Hurricane Andrew, primarily represented by a detailed damage map.

In 2005, Hurricane Katrina was another devastating hurricane that caused widespread damage, in particular to the city of New Orleans and its surrounding areas. van de Lindt et al. (2007) explored the performance of wood-frame structures in Hurricane Katrina. The authors thoroughly examined 27 case studies and made subsequent recommendations for better construction practices including more resilient timber design. In addition, Pistrika and Jonkman (2010) investigated the relationship of economic damage that occurred to residential buildings in Hurricane Katrina and flood characteristics. Furthermore, Ghosh et al. (2021) employed remote sensing technologies for damage assessments of Hurricane Katrina. Notably, Friedland (2009) introduced a wind and flood damage scale by using datasets describing the structural impacts of Hurricane Katrina and Hurricane Ike (2008).

Hurricane Sandy was yet another destructive hurricane that hit the United States as well as seven other countries in 2012. Many efforts were made to study the damage after this devastating hurricane such as Xian et al. (2015), in which the authors surveyed 380 structures in New Jersey after the hurricane, and worked on developing a quantitative analysis of the damage caused by hurricane surge using estimated damage percentages for each side of the building as well as non-structural component-level damage ratios. Hatzikyriakou and Lin

(2018) explored the impacts of storm flooding after Hurricane Sandy using aerial imagery to classify damage into five damage states that were then used in vulnerability analysis. Additionally, Tomiczek et al. (2017) developed a classification methodology for hurricane damage using Hurricane Sandy structural damage impacts on the coast of New Jersey.

Hurricane Harvey made landfall on Texas and Louisiana in 2017. Aghababaei et al. (2018) studied the structural damage that affected the city of Port Aransas, Texas, due to Hurricane Harvey. The authors showed the value of integrating both rapid, large-scale post-disaster damage assessments and detailed forensic evaluations of the damage. Moreover, Roueche et al. (2018) evaluated the performance of residential buildings affected by Hurricane Harvey. The authors also developed empirical fragility functions for single-family residential structures impacted by this hurricane.

In 2018, Hurricane Michael was another disastrous hurricane that made landfall in Florida. At the time, it was considered the fourth most powerful hurricane to strike the United States (Kijewski-Correa et al., 2018). Zhai and Peng (2020) used deep learning methods coupled with Google Street View to automate the process of damage assessments on Hurricane Michael. Kennedy et al. (2020) investigated the hazards that threatened the area as well as the the damage that occurred in Mexico Beach, Florida due to Hurricane Michael.

Amini and Memari (2020) provided a detailed literature review on the performance of coastal residential buildings under hurricane conditions. The authors explored the efforts of the natural hazards research community to mitigate the effects of natural hazards and the efforts made to understand these hazards better. Some notable works include White and of Chicago (1945), Pinelli et al. (2004), Taggart and van de Lindt (2009) Hamid et al. (2010), Park et al. (2017), and Masoomi et al. (2018).

In addition to hurricanes, major tornadoes hit mainland United States on a regular basis and are studied by the natural hazards engineering community. A few illustrative efforts are summarized here. Minor et al. (1977) developed fundamentals that show how buildings

interact with tornadoes in relation to wind speed and atmospheric pressure. Fujita (1993) studied the damage due to the Plainfield Tornado that occurred in 1990 and impacted Plainfield, Illinois. The authors presented photographs of the impacts of the storms and mapping of the storm damage. In 1999, a tornado outbreak struck Oklahoma and Marshall (2002) surveyed the performance of structures in the impacted area and presented a methodology to conduct tornado damage assessments. The Super Tuesday tornadoes that occurred in 2008, were deadly tornadoes that impacted the Southern United States. McMillan et al. (2008) deployed field surveys to document the damage from this tornado outbreak. Prevatt et al. (2012c) discussed observations made on the 2011 tornado outbreaks and the damage they caused. Prevatt et al. (2012a) studied the mechanisms of structural failure due to the Joplin, Missouri Tornado which occurred in 2011. Moreover, Prevatt et al. (2012b) performed damage surveys due to the 2011 Tuscaloosa, Alabama, and Joplin, Missouri tornadoes. Ramseyer et al. (2016) studied structural damages after the 2013 Moore, Oklahoma Tornado and proposed twelve code modifications to construct structural systems that can withstand an EF2 rating tornado. Roueche et al. (2017) created fragility functions that combined damage assessments of residential structures and a tornado wind field model that was conditioned on tree-fall. Strader et al. (2021) examined the Beauregard-Smith Station, Alabama, tornado and explained how Southeast US tornadoes typically unfold. Henderson et al. (2021) detailed damage observations from the tornadoes that struck Nashville, Tennessee in 2020. The authors also provided recommendations to improve the strength and performance of residential structures.

It can be seen from the discussion above that similar data types are typically collected in different post-windstorm reconnaissance missions. However, the scope, data collection methodologies, data formats, and presentation can all vary greatly from one reconnaissance mission to another, highlighting the need for a streamlined data enhancement and quality control protocol to standardize post-windstorm building performance datasets, which is one of the contributions of this work.

## 2.2 Post-windstorm Reconnaissance in Digital Format

The transition to an online or digital form for surveying damaged structures and collecting post-windstorm reconnaissance data considerably sped up the process and enabled the collection of a larger amount of data. An example of this is the Visualizing Impacts of Earthquakes With Satellites (VIEWS) system, which is a reconnaissance system that was developed by ImageCat. Inc in collaboration with the Multidisciplinary Center for Earthquake Engineering Research (MCEER) (Adams et al., 2005). This system uses satellite imagery of the affected areas pre- and post-disaster to assist the reconnaissance team in targeting areas for surveying based on the damage they sustained. It also contains a real-time GPS feed that helps in that process. Furthermore, it allows the surveyors to enter comments on any observed damage while in the field. During the field reconnaissance, geotagged photographs and videos are collected and later reviewed and analyzed. The VIEWS system has been used to survey damage due to the Super Tuesday tornado outbreak in 2008.

Moreover, Prevatt et al. (2012b) utilized a digital database of geotagged photographs and GIS damage maps when surveying damage due to tornadoes that impacted Tuscaloosa, Alabama and Joplin, Missouri. The authors made the datasets available to the research community on online interactive web portals.

CyberEye, developed by Kijewski-Correa et al. (2014), is yet another example of the post-windstorm data collection transitioning into a more digital format. It is an open-source collaborative virtual environment that combines post-disaster reconnaissance data collection with risk assessment that is used in emergency response planning. This system has a modular format that enables users to customize their dashboard to their preference. This platform was designed to approach post-windstorm reconnaissance from a more collaborative point of view to enhance the efforts of the members within the post-windstorm research community.

The Fulcrum platform (Spatial Networks, 2021) utilized by StEER and the RApp which is provided by NHERI RAPID facility (Berman et al., 2020) are examples of data collection platforms that allow the construction of collaborative geodatabases in real-time. They both

offer a mobile application that can be used on a personal device such as a phone or tablet, to which customized survey instruments can be deployed. Geotagged photographs, videos, and audio recordings can be embedded within each record and associated with the data collected via the survey instrument. Multiple users are able to simultaneously collect data and push the data in real-time (if data connectivity allows) to a web-platform for review and coordination. The collected data can further be reviewed and edited after field deployment has concluded. A built-in map is also available and shows the locations of the structures that were automatically geotagged during field reconnaissance. The Fulcrum application has an additional feature that shows the sustained damage intensity of the records on the available map. This is a great tool to visualize which areas sustained more damage than others in a particular event, which enables the user to further investigate these points. Additionally, the RApp contains user manuals for the equipment that the RAPID facility provides to the natural hazards community for use in their field reconnaissance missions.

### **2.3 NSF Structural Extreme Event Reconnaissance (StEER)**

StEER's datasets will be utilized throughout this thesis and to provide context for this data, a better understanding of StEER's organization and workflows is necessary. As introduced in Section 2.1.2, StEER is a collaborative network that consists of multidisciplinary engineers and other hazard research experts. StEER's primary goal is to collaboratively collect perishable postdisaster data that allows the natural hazards engineering community to study the performance of the built environment and analyze it in order to design structures and environments that are more resilient to natural hazards. This goal is achieved through coordinating the responses to these events with the participating members to ensure an efficient and prompt response. StEER's members participate in assessments using two methods, Field Assessment Structural Teams (FASTs) who collect perishable data using traditional field reconnaissance, and Virtual Assessment Structural Teams (VASTs) who collect data virtually from relevant sources and analyze it along with any field observations obtained



from FASTs. The raw data collected by FASTs is then processed by Data Librarians. Data Librarians are StEER members with less expertise, typically consisting of undergraduate and graduate students, that perform data enhancement and quality control.

StEER vigilantly monitors natural hazard events that are expected to occur. In the process of monitoring these events, the community of researchers and academic members of StEER communicate and exchange information about the expected event. A decision making process is executed to decide if an event warrants a response from StEER based on different factors. These factors include the expected severity and impact of the event as well as the lessons that could be learned from it. If the event is deemed significant, StEER coordinators begin planning for deployment of FASTs.

StEER deploys FASTs to perform Door-to-Door (D2D) building performance assessments, which can be performed either on-site or virtually using remote imaging sources. D2D assessments are conducted using multiple strategies and one of them is the Hazard Gradient Survey strategy used by StEER, which aligns with the scientific damage assessment methodologies outlined by Crandell and Kochkin (2005). In the Hazard Gradient Survey strategy, clusters of structures that lie across the hazard gradient are predefined. Then, the FAST surveys the structures in regular intervals to ensure that the final building performance dataset is not biased towards damaged structures and is a representative sample.

During their deployments, FASTs utilize numerous advanced tools to collect data. FASTs capture photographs that are automatically geotagged and uploaded directly into the Fulcrum application. Aerial imagery can also be collected by deploying Unmanned Aerial Systems (UAS). Moreover, street-level panoramic imagery is collected, processed, and uploaded into public mapping platforms such as Google Maps or Mapillary. Finally, point clouds may be collected using an imaging laser scanner for target areas of interest.

The major components of D2D assessments, which are performed by FASTs, consist of: metadata of the assessment (location, time/day, event, name of investigator, etc.), media

(photographs and audio), basic attributes (number of stories, year built, roof shape, etc.), structural attributes (structural system, roof system, foundation type, etc.), and damage assessments which consist of component level damage ratios, as well as an overall damage rating. During D2D assessments, due to time constraints and to collect data for as many structures as possible, FASTs typically only collect metadata, media, and a collection of specialized data related to load paths or data that needs special FAST forensic expertise. Because of this, the raw data collected by FASTs is usually non-uniform and in need of clean-up and standardization. Thus, after FASTs wrap up their D2D assessments, the resultant datasets need postprocessing to fill the gaps, which include basic and structural attributes, as well as clean-up to standardize the datasets in preparation for publishing. Here, again the need for a data enhancement and quality control protocol is highlighted. Such a protocol would describe and organize how FAST collected data can be used to develop high quality streamlined post-windstorm building performance datasets. This work details the development and execution of a data enhancement and quality control protocol which will be discussed in more details in Chapter 3.

## **2.4 Automating Damage Detection and Classification Using Machine Learning**

During the process of developing post-windstorm building performance datasets, the damage caused by a windstorm to the buildings in the dataset must be analyzed. This is a time consuming process where the various data sources discussed previously are used to gain a comprehensive understanding of the overall damage that impacted the structure that is then summarized and added to the dataset. With the emergence of promising modern technologies such as machine learning, research efforts focused on exploiting such technologies for the purposes of automating the damage detection and classification steps. Such efforts will be the focus of this section.

### 2.4.1 Notable Work on Automating Damage detection

Thomas (2012) explained how some of the previous work done on the topic of damage detection is pixel-based and how before and after disaster images are analyzed. For example, Adams et al. (2004) used change detection techniques including edge detection and texture analysis of pre- and post- earthquake imagery. Similarly, Lakshminarasimhan (2004) used change detection to determine the extent of damage to various segments from labeled pre-disaster images. The damage detection performed on these segments was found by implementing grey-level intensity as well as edge detection and the connectivity of pixels. Additionally, Sirmacek and Unsalan (2009) discussed automated damage detection using extracted shadow segments. Kakooei and Baleghi (2017) proposed a framework combining damage detection of oblique and aerial imagery.

### 2.4.2 Machine Learning and Deep Learning Technologies in Damage Classification

Machine learning and deep learning technologies have been a topic of interest in the natural hazards research community recently. This interest is motivated by the huge potential such technologies offer, for example, to build and train models that can rapidly classify structures into predefined damage categories with high accuracy. Thomas et al. (2014) used supervised classification by implementing the random forest algorithm on post-hurricane high-resolution aerial imagery to predict missing roof shingles, collapsed roofs, and any cavities that occurred due to structural roof damage. Li et al. (2018b) built a framework that utilizes unsupervised learning with convolutional autoencoders to classify post-hurricane aerial imagery, followed by a process to refine the classification using convolutional neural networks (CNN). Vetrivel et al. (2016) used a visual-bag-of-words model combined with the use of supervised learning algorithms to classify image patches from post-disaster oblique imagery into two categories, damaged and undamaged. Bialas et al. (2016) made a comparison of object-based classification and pixel based machine learning algorithms on post-earthquake

high resolution aerial imagery. The authors found that object-based methods give better results than pixel-based methods of classification. Nia and Mori (2017) used three neural networks, where a regressor was employed to summarize the features that were extracted using the three neural networks, to finally give a damage assessment. Yeum et al. (2018) explored the use of CNNs to predict collapse and spalling in structures subjected to post-disaster damage. Zhai and Peng (2020) demonstrated the use of Google Street View imagery to classify post-disaster damage using deep learning algorithms while comparing the performance of Google Street View imagery to remote sensing aerial imagery. The authors found that Street View imagery performed better particularly when the damage observed was minor, while aerial imagery was more accurate when the damage was severe. Radhika et al. (2015) explored the use of an artificial neural network combined with the use of support vector machines algorithm, after the buildings were detected using wavelets pattern recognition, to detect roof damage.

### **2.4.3 Automating the Detection of Building Attributes**

Post-windstorm reconnaissance datasets are enhanced by the addition of building attributes that describe the structures that were surveyed, such as number of stories, roof slope, roof shape, presence and location of fenestration and fenestration protection. This additional data can be particularly helpful in analyzing the effects of extreme wind events within the context of attributes that influence or are correlated with structural performance (Egnew et al., 2018; Crawford et al., 2020; Fronstin et al., 1994; Silva et al., 2008). These analyses include developing fragility functions which can be seen in Massarra et al. (2020), damage models shown in Pinelli et al. (2004), and developing damage assessment methodologies as shown in Friedland (2009).

Efforts to automate acquiring this data using artificial intelligence and deep learning approaches have been made. An example of this is Building Recognition using AI at Large-Scale (BRAILS) (Wang et al., 2021; Yu et al., 2019b), which is a framework that implements

the use of deep learning to develop a building inventory database by extracting building attributes from satellite and street view imagery. The framework is broken into four steps, first, a region of interest is selected and basic building attributes would need to be collected from tax websites such as addresses, number of stories and the exterior siding of a structure. Next, the addresses are used to find the coordinates using Google’s application programming interface (API). In the next step, the coordinates are used to collect satellite and street view images that are associated with each of the addresses provided. After that, a CNN is utilized to predict building features including roof shape, occupancy type, decade of construction, and predicting if the structure is soft story. Finally, the building inventory database is enhanced using Spacial Uncertainty Research Framework (SURF), that was developed by Wang (2019), and is a package built in the Python programming language that performs spatial uncertainty analysis and employs the use of machine learning to predict missing attributes.

Kang et al. (2018) worked on identifying individual building attributes from street view or satellite imagery utilizing CNNs to classify building occupancy type using street view and satellite imagery. Moreover, Li et al. (2018a) employed the use of CNNs for feature extraction combined with support vector regression to estimate the year of construction for buildings from street view imagery. Furthermore, Kong and Fan (2020) performed facade parsing using Mapillary imagery, where a pipeline of CNNs was utilized to classify the segments of facades of buildings. The majority of these applications lack the needed accuracy to viably replace human-based approaches but have the potential through future improvements.

## **2.5 This Thesis**

As can be seen from the literature review in this chapter, significant advancements in the field of post-windstorm reconnaissance have been made, ranging from using different types of image capturing techniques to record and collect post-windstorm data, to the exploration of modern computer technologies to detect roof damages. However, many challenges still need

to be addresses and there is ample space for further improvements. The following details the contributions of this work:

- Development of a post-windstorm data enhancement and quality control protocol that details how the raw data collected from post-windstorm reconnaissance missions can be transformed into streamlined high-quality building performance datasets that can be reused in various analyses.
- Development of a training regimen to prepare engineering students to perform the data enhancement and quality control process according to the developed protocol.
- Development of an automation framework that utilizes various modern technologies, such as web scraping and machine learning, to automate various parts of the data enhancement and quality control process.

## 2.6 Chapter Summary

This chapter walks through the developments of post-windstorm reconnaissance throughout the years, highlighting significant contributions to the collection of this data and showcasing the importance of, and key findings from, analyzing such data. The developments discussed range from the use of traditional data collection techniques, to the utilization of advanced modern technologies such as machine learning. The efforts covered in this chapter include both organizational as well as academic contributions, with a focus on StEER's efforts as the work covered in this thesis incorporates their datasets. This literature review highlights the need for developing a protocol to enhance building performance datasets collected in reconnaissance that can be reused in various analyses, which can ultimately lead to increasing structural resiliency. This chapter provides context for the following chapters where the development of a data enhancement and quality control protocol is discussed, and modern tools to automate parts of the data enhancement and quality control process are explored.

## Chapter 3

### Data Enhancement and Quality Control Protocol

As introduced in Section 2.3, one of the primary products StEER produces following wind events is a building performance dataset that documents the performance of multiple buildings affected by the hazard. These datasets ideally contain the damage experienced by a building, along with the key contextual information that would help explain why a given building performed as it did, in addition to the metadata of the investigation itself. This information is primarily packaged within a standardized survey instrument that, for building assessments following windstorms, contains over 100 individual fields. Table 3.1 provides a summary of major classes of information that are present. The entire list of fields is available in Appendix.A and is described in more detail in Kijewski-Correa et al. (2021). Only a fraction of these fields, deemed field-critical or required in the StEER FAST handbooks and in the StEER Building – US (Windstorm) Fulcrum app, are typically filled out on-site by the FAST. Most of the fields must be added post-mortem and even those fields that were entered on-site, must be checked for accuracy and validity before final publication. This work develops a protocol to enhance the raw data and prepare it for final publication through a Data Enhancement and Quality Control (DE/QC) process, which will be the primary focus of this chapter.

Figure 3.1 illustrates the scale of these efforts. Since 2018, StEER FAST/VAST have combined to generate 5492 records of building performance in twelve different windstorm events. Each of these records must undergo the DE/QC process described in the following sections before the data is published to the DesignSafe-CI.

Table 3.1: The major data classes in StEER’s datasets.

Major Data Class	Description
Metadata	Information such as location, project name, data/time,etc
Basic Information	Information such as name of investigator, Assessment type, etc
Media Attachments	Photographs and audio
Overall Damage Assessments	Wind damage rating, surge damage rating, etc
Building Attributes	Descriptive building attributes such as number of stories, year-built, etc
Structural Attributes	Structural components of the building such as building type
Wind-Induced Damage Levels	Component-level damage ratios due to wind impacts
Surge-Induced Damage Levels	Damage to components impacted by storm surge
Quality Control Tracking	Quality control codes that indicated the stage the record is at

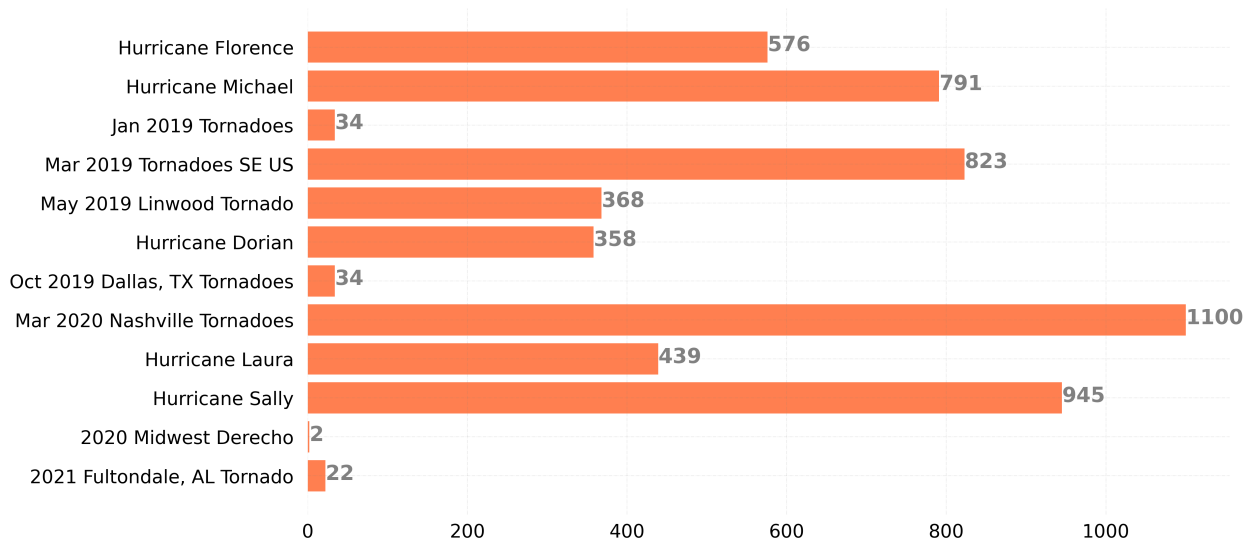


Figure 3.1: Distribution of StEER’s records per windstorm that was surveyed by FAST.

### 3.1 Framework for Data Enhancement and Quality Control

Figure 3.2 summarizes the framework that was developed in this work for StEER’s data enhancement and quality control. The data enhancement process begins after the FAST collects an initial dataset in field reconnaissance that consists of geotagged records containing mostly metadata, photographs, audio (when available), and notes in text format. Before data enhancement begins, pre-processing of the raw records is performed to ensure that the remaining steps in the process go smoothly. Next, the data enhancement process is



performed, and is followed by quality control. Finally, the data is published to DesignSafe-CI. The following sections will cover the StEER Windstorm Building Assessment Survey Instrument and discuss in detail each component of the built DEQC framework.

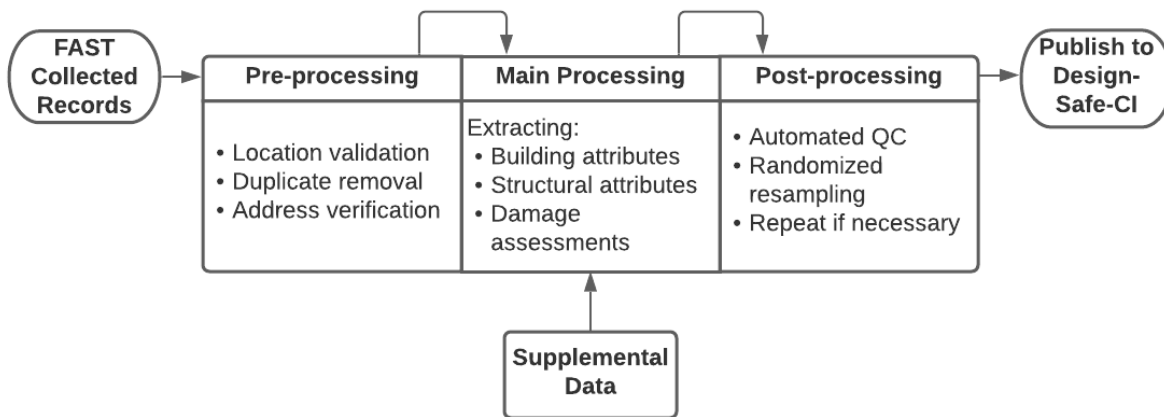


Figure 3.2: Flowchart of the framework for data enhancement and quality control.

### 3.2 Pre-processing Tasks and Considerations

Before the main data enhancement process can begin, the location of the surveyed structures must be validated, and corrected if needed, using a geographic information system (GIS) software for a manual approach, or using an automated script, for a less labor-intensive approach. If GIS software is used and building footprints are available from county tax records, the process is performed by simply employing a point in polygon query. However, if there are no building footprints available, a manual visual check can be performed using a base layer, such as OpenStreetMap, and a layer containing pins of the surveyed structures represented by points. Validation is then performed by manually checking that every point is over the correct building. An example of this can be seen in Fig3.3 where 3.3a and 3.3c represent the locations before location validation, where the pins are overlaid on an OpenStreetMap basemap, and on NOAA aerial imagery, respectively. 3.3b and 3.3d show the same records in 3.3a and 3.3c after the location validation process. The benefit of using

more than one base map can be seen here as well, where the OpenStreetMap basemap provides building footprints for some buildings, while the NOAA aerial imagery shows the damaged buildings for perspective.

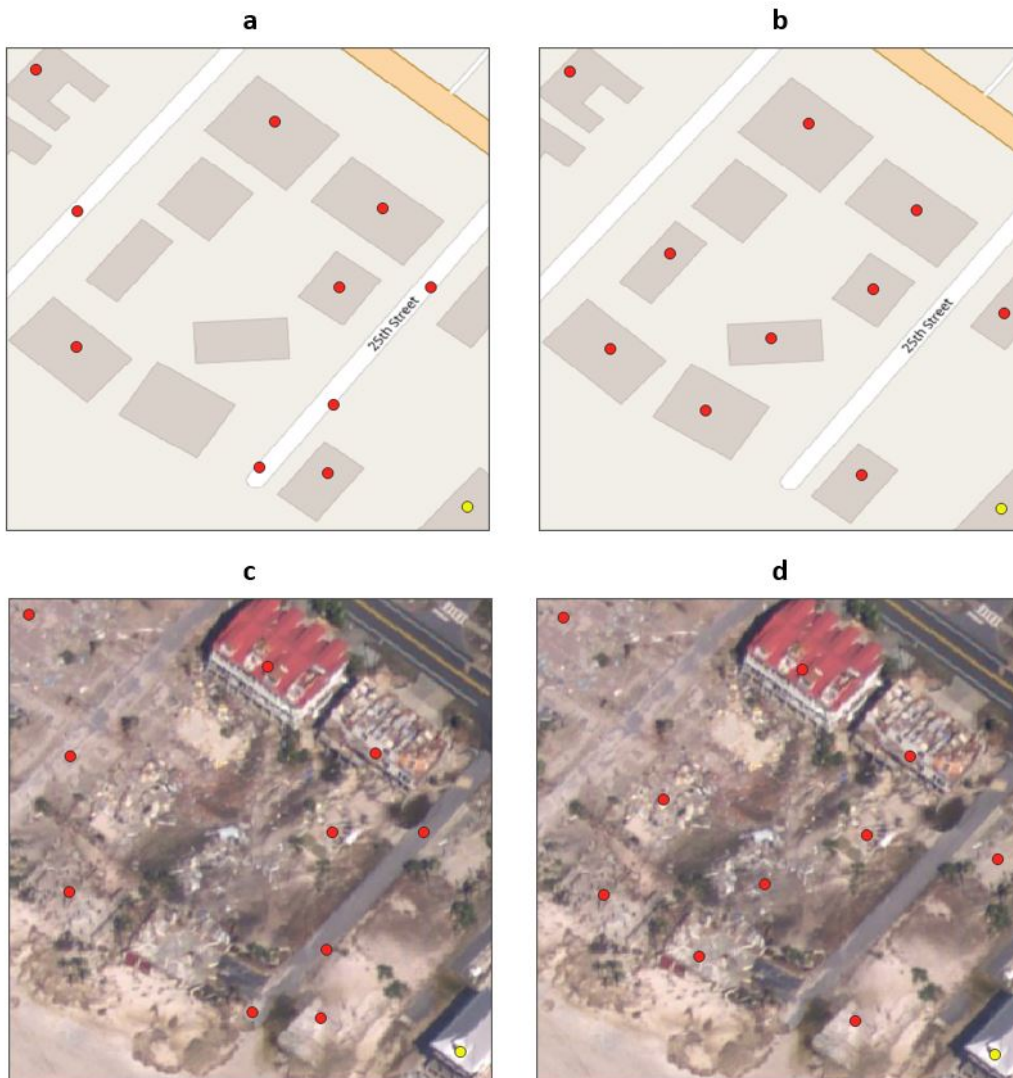


Figure 3.3: Location validation for Hurricane Michael Dataset. **a** and **c** represent the locations of the records obtained from field reconnaissance before correction overlaid on OpenStreetMap basemap and NOAA aerial imagery respectively. **b** and **d** represent the locations of the records after placing them over the correct buildings overlaid over OpenStreetMap basemap and NOAA aerial imagery respectively.

The easier approach is to use building footprints from county tax records, or from open sources such as AI-generated footprints available from Microsoft (Microsoft, 2018), along with a script that employs a point in polygon function to flag records that do not fall

within their respective footprint polygon. Only the outliers resulting from this process would then need to be manually corrected, which significantly reduces the manual effort needed compared with the other approaches. Searching for duplicates is another pre-processing step that is usually performed jointly with the location validation process. Sometimes multiple records of the same assessment will get created during field reconnaissance. It is important to consolidate the information in these records into one record. Additionally, the addresses can be verified from public county appraisals or by employing a script that uses Google API to find the addresses from the coordinates obtained from the field.

### **3.3 Main Processing Tasks and Considerations**

This section will focus on the data enhancement process that the Data Librarians perform and will follow the order of the major data classes that are present in StEER Building - US(windstorm) application. The media attachments which consist of photographs and audio recordings captured by the FAST are examined carefully.

The next step is to define the building attributes associated with the record. These can be extracted from the media attachments, pre- and post-event street view panoramic imagery, point clouds (if available), and Pictometry imagery (if available).

Additionally, to reduce the number of manual tasks that must be performed for each record, where available, public county appraisals in GIS format are used to extract the basic attributes of the structures. The county appraisals are generally reliable as they are official records. However, the data still is subjected to validation through visual inspection and other data sources as well. Before using the county data, the location validation process must be performed to ensure the successful merging of the county data with the dataset that has been built up until now. Merging the data is conducted using a spatial join on GIS software, which is a process that joins attributes from one layer to another based on spatial relationships. At this point in the process, the result is a dataset of location-validated records with photos of the sampled structures from D2D assessments along with any information

collected and entered by the field investigators, and the basic attributes of the structures that were obtained from the public county records.

Although the county data is reliable, care must be taken when using the data due to the heavy use of abbreviations and the inconsistency of the abbreviations across different counties. If it is not clear what an abbreviation is referring to, the data associated with that abbreviation is not used. This is done through a cleanup process where the unclear abbreviations are discarded while the clear ones are mapped to the corresponding available options used by StEER in their online form. After the cleanup process is conducted, the data is ready to be imported back to the Fulcrum platform. Table 3.2 shows an example of data extracted from county records before cleanup, where both blanks and abbreviations are present. Table 3.3 shows how the data would be cleaned up for use in StEER’s dataset. Here, values such as "FRAME" and "TYPICAL" are discarded because either their meaning is ambiguous or unknown, or, as in the case of "FRAME", it refers to a different component of the structure.

Table 3.2: Examples of data found in county records before cleanup process

<b>Structure</b>	<b>Exterior</b>	<b>Foundation</b>	<b>Stories</b>	<b>Year Built</b>
SINGLE FAMILY	FRAME	FULL BSMT	1.5 STORY	1900
DUPLEX	BRICK	SLAB	1 STORY	1985
-	BRICK	TYPICAL	COM 1 STY	1935

Table 3.3: Data extracted from county data after cleanup

<b>Occupancy</b>	<b>Wall Cladding</b>	<b>Foundation Type</b>	<b>Number of Stories</b>	<b>Year Built</b>
Residential, Single Family	-	Unreinforced masonry stem wall	1.5	1900
Residential, Multi-Family Homes	Brick	Slab-on-grade	1	1985
-	Brick	-	1	1935

Next, the structural attributes are defined. These attributes are extracted similarly to the building attributes. Some of these attributes rely on information provided by the FASTs through voice recordings or text notes entered into Fulcrum. For undamaged structures, the structural attributes are often unknown. Some information can be inferred from similar, nearby structures if more detailed assessments were performed there, or damage uncovered the structural system. But often individual connection types are not discernible at the DE/QC stage if they were not defined by the FAST in the field.

Finally, component-level damage ratios are estimated using post-event imagery. Street-level panoramic imagery is particularly helpful for these assessments because they provide imagery to sides of the structure that might not have been visible in D2D assessments. Roof damage ratios, which include roof cover damage, roof substrate damage, and roof structure damage can be estimated using a combination of aerial imagery, street-level panoramic imagery, and D2D assessments. Each of these sources provides a different viewpoint that can be used to get a more accurate assessment overall. After all component-level damage ratios are completed, the overall damage ratings can be determined based on the ratios that were entered.

To track the progress of the data enhancement and quality process discussed in this section, a QC code is assigned to each record. Table 3.4 shows the criteria for assigning QC codes to records, based on StEER's DE/QC handbook. QC code 1 indicates that the record contains a verified address and location. Once the record contains an overall damage rating and basic building attributes such as year-built and occupancy, the record is marked as QC code 2. QC code 3 is reserved for records that contain damage assessments where all component level damage ratios are filled in with as much accuracy as possible. Next, QC code 4 consists of records that contain data in field-priority fields such as connections and roof sheathing type. The data in these fields are more difficult to extract from images and need an individual with more expertise on the topic to fill them in. Records that nominally reached a given stage but because of missing data or other circumstances have higher uncertainty

Table 3.4: StEER’s QC codes, which are used to track the data enhancement and quality control progress. Data Librarians will enter these codes based on the criteria shown in the table.

<b>Code</b>	<b>Meaning</b>
<b>1</b>	Location and address have been verified. The record location is positioned directly over the building that is the focus of the investigation.
<b>1e</b>	The location and address have not been verified due to an error or uncertainty. The exact location or address of the building is not able to be confirmed. If a record is at 1e, it may not be possible or worthwhile to advance into additional stages.
<b>2</b>	Stage 1 has been completed, and the minimum information for a completed assessment has been verified or added. For example, the correct building type is assigned, overall damage ratings are confirmed to be in agreement with the standard quantitative guidelines, and basic building attributes such as year built are present.
<b>2e</b>	Stage 1 has been completed but there is insufficient information to meet the minimum data standards for a complete assessment, or there is considerable uncertainty in assignment of one or more critical fields. If a record is at 2e, it may not be possible or worthwhile to advance into additional stages.
<b>3</b>	Stage 2 and below has been completed. The majority of Stage 3 fields as identified in Table 2 have been completed and validated with reasonable confidence in accuracy and precision.
<b>3e</b>	Stage 2 and below has been completed, and some Stage 3 fields have been completed, but lack of data (e.g., only 2 sides of the structure are visible) prevents the assessment from being completed without undue uncertainty. It could be helpful to consider the tornado/hurricane path and the wind direction to determine if the lack of visibility of all sides is preventing adequate damage assessment.
<b>4</b>	Stage 3 and below has been completed, and more detailed forensic fields such as roof-to-wall connection type and foundation anchorage type have been added and verified. Most fields are filled in for the record with reasonable levels of confidence. Most records will not get to Stage 4. Records that do get to Stage 4 may be good candidates for detailed case studies.
<b>5</b>	Final QC validation has been completed with automated and manual checks. The record is ready to be published in DesignSafe.
<b>5e</b>	Final QC validation has been completed but errors have been identified that need to be corrected manually. Once the errors are resolved, the code should be changed to “5”.

than usual, are tagged with an additional ”e” after the QC code (1e,2e,3e,etc.). This flags these records so that they could be refined or removed before performing analyses if needed.

### 3.3.1 Extension of DE/QC to Virtual Assessments

Many aspects of the DE/QC process outlined above could also be applied as fully virtual assessments, wherein no or very few on-site assessments are performed, and information for the records are sourced almost entirely from remote imagery. This extension of the DE/QC process became critically important in 2020, with the onset of the COVID-19 pandemic. The pandemic limited the scope and methods available to field reconnaissance teams during the most active year of land-falling hurricanes in recorded history, necessitating alternative approaches to still learn from the disasters and generate building performance datasets. A hybrid approach was developed in which FAST prioritize remote imagery capture technologies (UAV, street-level panoramas) in tandem with a few detailed forensic evaluations in select clusters to characterize the structural load path. Blank records are then created in Fulcrum in clusters of similar structures across the hazard gradient where at least street-level panoramas were captured, but ideally where some UAV flights and forensic assessments were also conducted, VAST and Data Librarians then nominally follow the DE/QC process, using information from the few detailed forensic studies to inform defining aspects of the structural load path that would not be visible in the remote imagery.

This approach was first piloted with Hurricane Laura (2020) as described in Roueche et al. (2021). Following landfall, a semi-virtual damage survey was performed by StEER using the Rapid Survey strategy to assess the damage due to Hurricane Laura which occurred in 2020. The virtual assessment was supplemented with street-level panoramic imagery taken by a small FAST that was able to capture it, UAS imagery also captured by the FAST, and aerial imagery publicly available by NOAA. The key to the success of this semi-virtual approach was the essential information provided by the FAST to the Data Librarians about connection and construction details. The FAST was able to collect information about a representative number of structures that were present in each cluster, which enabled the Data Librarians to make some assumptions about similar data points such as structures consisting of similar construction, year of construction, occupancy, etc. The data enhancement process

for this dataset included a damage survey aspect as well. As with the Hazard Gradient Survey strategy followed by StEER in a more traditional form of field reconnaissance, this Rapid Survey strategy followed a similar procedure for sampling the structures in each cluster. A large amount of street-level panoramic imagery was collected from a wide range of areas affected by Hurricane Laura. Clusters were chosen from the areas covered by street-level panoramic imagery and to get a representative sample that can be used for statistical analyses later on, every other and in some cases, every third structure was selected from the cluster to create a representative unbiased sample. Screen captures from the street-level panoramic imagery were taken from all of the visible sides of each surveyed structure, similarly to the procedure that is used when D2D assessments are feasible. After that, the usual procedure for data enhancement was followed. This new strategy that was developed due to challenges present from the global pandemic, enabled surveying the damage due to the windstorm and collecting data related to the damage while maintaining the safety of all individuals involved in the process.

Employing this new strategy showed how the data enhancement and quality control processes can easily be transitioned to a fully virtual assessment when needed. When this fully virtual assessment is supplemented with a few detailed on-site assessments to characterize major clusters, it shows even more potential.

### **3.3.2 StEER Windstorm Building Assessment Survey Instrument**

The StEER online damage assessment forms are currently hosted on the Fulcrum platform (Spatial Networks, 2021). The damage assessments made by StEER are organized by natural hazard into individual applications. There are three StEER windstorm applications: an application focused on building assessments, another dedicated to nonbuilding structures, and a third dedicated to capturing hazard intensity observations. The focus in this thesis will be on the StEER Building - US(windstrom) application. Part of the contributions of



this thesis include updating and modifying this application to increase efficiency and introduce new features, such as adding direct links to Google maps and other types of available imagery.

When one of StEER’s applications is accessed, the user will see a collection of all available records. Each record is identified by an automatically generated unique string, called the Record ID. By default, all the records appear in the application in a table format shown in Figure 3.4.

The screenshot shows the 'archive\_StEER Building - US (windstorm)' application interface. It features a search bar at the top with '4219 records' and a 'SAVE VIEW' button. On the left, there are 'QUICK FILTERS' for 'Record Updated' (with options like All, Today, Yesterday, Last 7 days, Last 30 days, This Month, Last Month, and Specific Range) and 'Status' (with options like Destroyed, Severe, Moderate, Minor, Undamaged, and Unassigned). The main table displays the following data:

Record ID	Damage State	Title	Updated	Project
5c33b8e8-2cb2-468d-adda-fbc2717cdb52	Minor	David Prevatt, 2305 SONDRRA CT Southport FL 32409 USA	10/8/2020, 12:54:23 PM	Hurricane Michael
5e9424ad-161b-42c0-8817-0855cee4e943	Severe	John Cleary, 1808 EVERITT AVE Panama City FL 32405 USA	10/8/2020, 12:54:08 PM	Hurricane Michael
5bca4616-afe4-462b-b6af-ba23c7490b39	Minor	David Prevatt, 20496 NE LAMBERT ST Blountstown FL 32424 USA	10/8/2020, 12:53:43 PM	Hurricane Michael
3d527a00-400e-439e-930c-87e4be96e09e	Minor	David Prevatt, 17565 CHARLEY JOHNS ST NE Blountstown FL 32424 USA	10/8/2020, 12:53:32 PM	Hurricane Michael
88788a7b-994e-46b7-a44f-1da99673b27e	Moderate	Keith Cullum, 7326 MARY JO AVE Southport FL 32456 USA	10/8/2020, 12:53:07 PM	Hurricane Michael
90cd760a-cfd7-48bc-b9c1-783f885e8f39	Destroyed	Jean-Paul Pinelli, 9105 HWY 98 Port Saint Joe FL 32456 USA	10/8/2020, 12:52:55 PM	Hurricane Michael
b26b866a-e460-47ba-8034-c66fa20f1536	Minor	Jean-Paul Pinelli, 20569 LAMBERT AVE NE Blountstown FL 32424 USA	10/8/2020, 12:52:20 PM	Hurricane Michael
f14c636c-53a6-4110-a2c3-03b4685c72e8	Minor	Jean-Paul Pinelli, 406 ARIZONA DR Mexico Beach FL 32456	10/8/2020, 12:51:14 PM	Hurricane Michael

Figure 3.4: Inside look at StEER’s archive windstorm application showing the table view in Fulcrum.

Each record in the windstorm StEER applications consists of multiple main sections. This includes sections for: overall damage assessments due to the hazards present, (wind, tree fall, surge, etc.), in the event of interest, building attributes, structural attributes, and sections dedicated to damage assessments. These sections will be discussed in more detail below. Additionally, an example of an open record is shown in Fig 3.5.

There is a map available at the top of each open record, which displays the location of the record and provides options to tailor the base maps to one’s liking, and includes a

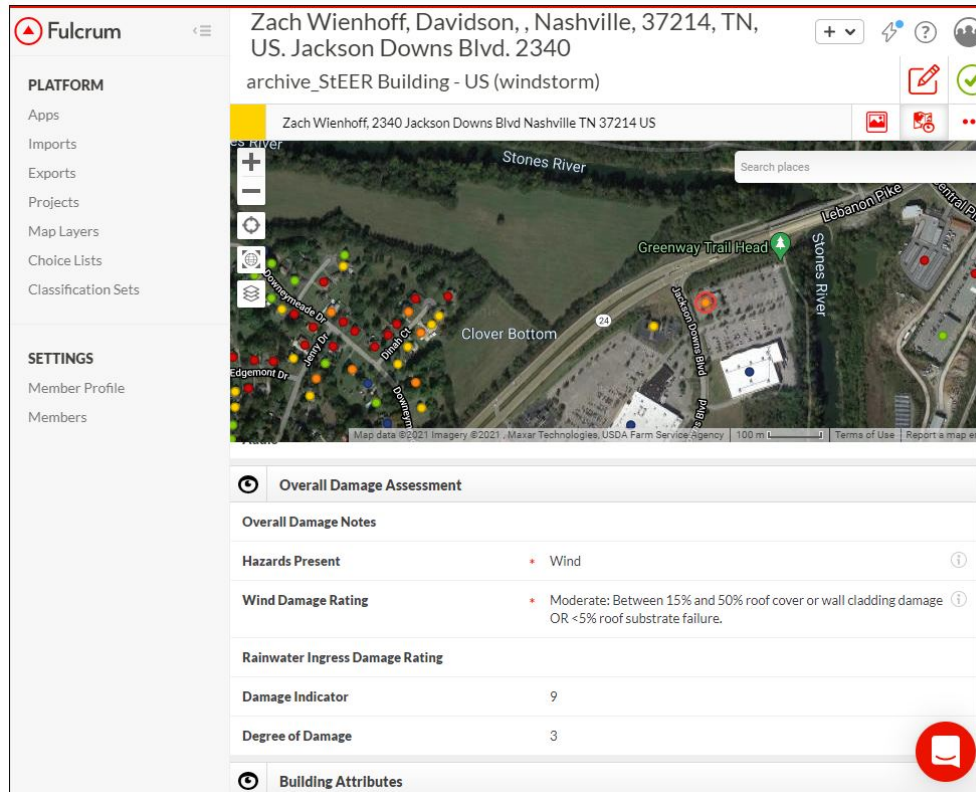


Figure 3.5: An example of a record from the Nashville Tornado that occurred in 2020. The map shows the location of the surveyed structure and below are some of the fields present in this record.

zooming functionality that enables the user to control what is visible within the frame of the map. The map, as well as the available options, are shown in Figure 3.5. Below the map is a section dedicated to metadata, which consists of data related to the creation and updating of the record. Next is a section containing basic information about the assessment, such as the name of the investigator that performed the assessment and the assessment type.

The next section in each record is the media attachments which contains any photographs, either taken in person when field reconnaissance is possible, or screenshots taken from satellite or street-level panoramic imagery. Additionally, audio recordings that were captured by the FAST are in this section as well. The following section is overall damage assessments which consists of a list of hazards that were present, including wind damage rating, surge damage rating, rain ingress damage rating, and an EF damage indicator and degree of damage reserved for tornadoes Marshall et al. (2004). The wind damage rating,

shown in Figure 3.6, is a quantitative measure of the damage that occurred due to the wind and is based on the rating that was constructed by Vickery et al. (2006). The surge damage rating is a quantitative measure of the damage that occurred due to storm surge, and the surge damage rating developed by Friedland (2009) is used as the criteria for this damage rating.






Damage State [1]	Short Description	Illustrative Example	Presence or Extent of Failure in:					Fascia and/or soffit
			Roof or Wall cover	Window or door	Roof or Wall substrate	Roof struct.	Wall structure [2]	
0 No damage	No visible exterior damage		0%	No	No	No	No	No
1 Minor damage	Damage confined to envelope		> 0% and ≤ 15%	1	No	No	No	≤ 20%
2 Moderate damage	Load path preserved, but significant repairs required		> 15% and ≤ 50%	> 1 and < the larger of 3 and 20%	1 to 3 panels	No	No	> 20%
3 Severe Damage	Major impacts to structural load path		> 50%	> the larger of 3 and 20% and ≤ 50%	> 3 and ≤ 25%	≤ 15%	No	
4 Destroyed	Total loss. Structural load path compromised beyond repair.		> 50%	> 50%	> 25%	> 15%	Yes	
Notes: [1] A building is considered to be in the damage state if any of the shaded damage indicators in the corresponding row are observed. [2] Wall structure refers to walls in living area only. The ground floor of elevated structures often have breakaway walls that can be easily damaged by storm surge. This damage should be ignored in assigning the overall damage rating.								

Figure 3.6: StEER's Wind Damage Rating criteria which can be found in the DE/QC handbook.

A building attributes section is next, which consists of identifying features about the building, including the address of the structure, year-built, occupancy, etc. The occupancy

field is based on the occupancy definitions in the 2015 International Building Code (IBC) with some modifications to suit the nature of the collected data.

Next is the structural attributes section, which focuses on structural features of the structure. An example of fields in this section is building type, which refers to the structural classification that the structure falls under. The building type field follows the classifications outlined in FEMA (2000). Furthermore, this section consists of multiple field-priority fields that are more suited for the FAST to fill in that define specific connections within the structural load path.

Next are two damage assessment sections. The first damage assessment section focuses on component-level damage ratios induced by wind, while the second damage assessment section focuses on surge damage details. These sections are of particular importance because they allow the use of a damage rating with different quantitative criteria than the overall damage ratings used by StEER.

StEER used ATC (2005a,b); Friedland (2009); Woolpert (2006), and FEMA MAT standard operating procedures (FEMA, 2008) as building blocks to develop the fields of the windstorm applications.

The geotagged records in these applications are the main workspace that the Data Librarians work on, where they enter the data they collect and make their damage assessments of the structures. Several data sources are available for the Data Librarians to work with during the data enhancement process. These data sources will be discussed in the following section.

### **3.3.3 Supplemental Data Sources**

Depending on what strategy was implemented by the FAST in the field reconnaissance, and what other organizations responded to the event, a variety of data may be available to the Data Librarians to analyze and extract key information from. Many of these data sources are intentionally overlapping, allowing the Data Librarians to have greater confidence that

the data being extracted is conclusive and accurate. Data sources can be broadly classified into the categories shown in Table 3.5.

Table 3.5: Main data types available to use in data enhancement along with their sources.

Main Data Types	Source
Public county appraisals	County GIS platforms, realtor websites
Pre-event street level panoramas	Google Maps
Post-event street level panoramas	Captured by FAST and uploaded to Mapillary or Google Maps
Pre-event aerial imagery	Google satellite imagery, Pictometry
Post-event aerial imagery	NOAA aerial imagery, Pictometry
FAST data in Fulcrum	FAST D2D assessments
FAST point clouds	FAST deployments

Public county appraisals are used to extract the address, exterior material of the building or roof, the material the structural system is made of, and the foundation type. If the files were not obtained from the county in shapefile format, the data is extracted manually from tax county websites. Also, realty websites such as Zillow are used similarly and comparable data can be extracted from them as well.

If street-level panoramic imagery was collected in field reconnaissance, it serves as a great resource for the data enhancement process. The ability to "drive" by each structure back and forth and check every visible angle is very helpful. If photographs of the structure contained in the door-to-door investigations were not sufficient to make an assessment, or not all sides of the structure were photographed, street-level panoramic imagery could fill in the gaps. The majority of the street-level panoramic imagery used in this study were captured at a resolution of 11000x5500 pixels, which provided sufficient resolution for zooming in on details of the structure.

Publicly available post-disaster satellite imagery collected by other teams, such as NOAA aerial imagery, is very helpful in the data enhancement process. Imagery such as this gives a viewpoint that might be absent from other available resources particularly for the roofs of the structures that might not be visible in ground-level imagery sources. Figure 3.7 shows an example of this, where if the assessment relied only on the D2D photograph

in Figure 3.7a the assessment would be inaccurate if the roof was assessed as undamaged. Figure 3.7b shows the roof of the same structure in NOAA aerial imagery, and it can be seen that the roof sustained damage as indicated by the blue tarp covering part of it, as tarps are typically used when damage is present. NOAA aerial imagery is taken within few days of the event and depicts the impacts before repair and cleanup processes start. This provides a more accurate assessment of the damage to the roof and an overall more accurate damage assessment of the whole structure.

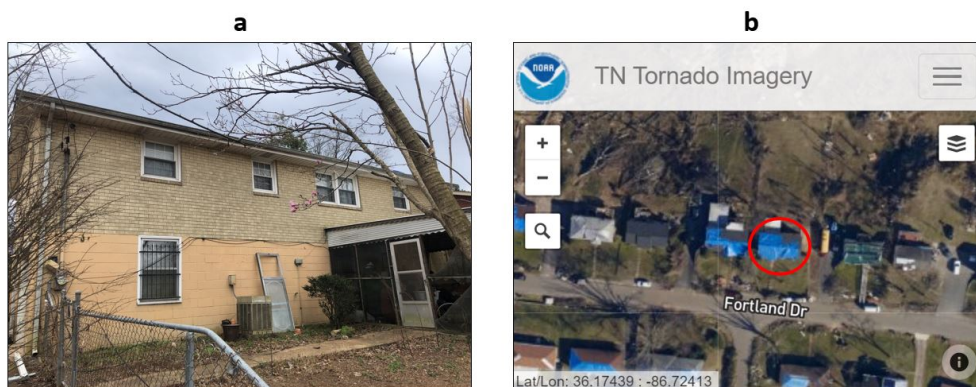


Figure 3.7: An example of a structure impacted by Nashville Tornadoes 2020 where the roof is not visible from the D2D photograph but is visible in the NOAA aerial imagery. a) A structure in a photograph from StEER D2D assessments where the roof is not visible. b) The roof of the same structure in NOAA aerial imagery where it can be seen that the roof is damaged.

Point clouds collected using UAS provide a similar viewpoint to satellite imagery but with an added advantage of the ability to rotate the viewpoint and view the structures from multiple sides, and the ability to measure 3D properties using the point clouds. They are also not limited to the orthogonal imagery view, oblique views of the structures are available as well through point clouds. Examples of this imagery can be seen in Figure 3.8, where Figure 3.8a shows imagery of structures impacted by Hurricane Laura in 2020, and Figure 3.8b shows imagery of structures impacted by Hurricane Michael in 2018.

Another source of data is imagery from Pictometry (EagleView Technologies, 2021) when available. Pictometry provides aerial imagery which includes side views of the structures from multiple directions as well as orthogonal imagery of the roofs. Imagery before





Figure 3.8: Point Clouds captured by the FAST in field reconnaissance. a) Hurricane Laura in (2020) imagery, b) Hurricane Michael in (2018).

the windstorm is available sometimes as well, which helps identify some of the building attributes such as roof slope or roof cover type, this is helpful when the damage does not allow these attributes to be identified. An example of Pictometry imagery of a coastal residential structure that was damaged by Hurricane Michael in 2018 can be seen in Figure 3.9. Figure 3.9a represents the front and left sides of the structure and Figure 3.9b represents the top and left sides of the structure.



Figure 3.9: Pictometry imagery of a coastal residential structure impacted by Hurricane Michael in 2018. a) Front and right sides of the structure, b) top and left sides of the structure.

The final data source is the data collected in the door-to-door assessments, which is one of the main sources of data for the Data Librarians. This data includes photographs taken on-site, audio recordings by the FAST members during field reconnaissance, and notes on specific field observations such as load path observations or connections and details that require more expertise which the FAST would be more knowledgeable about. Figure 3.10 represents an example of data sources that were available for Data Librarians for a record from the Nashville Tornado dataset that occurred in 2020. For this particular record, street-level panoramic imagery was available, NOAA aerial imagery, realty website information, and the photographs and audio captured by the FAST on Fulcrum.

After the data enhancement process is performed on the dataset, a quality control process is performed to detect errors and increase the accuracy of the data.

#### **3.3.4 Logistics of the DE/QC Process**

In order to perform the DE/QC process for a set of records, two main components are needed: First, a platform that enables the storing, editing, and manipulation of the data to be processed, and second, trained persons that can use the platform and are familiar with the DE/QC process. For the DE/QC framework discussed here, the primary platform of choice is the Fulcrum platform, and the trained persons are the Data Librarians, which are discussed in Section 3.5.

As discussed in Section 3.3.2, the StEER online damage assessment forms are currently hosted on the Fulcrum platform. Fulcrum's landing page can be seen in Figure3.11, which shows StEER's windstorm applications.

When FAST members are in the field, they use the StEER Building - US(windstorm) application on the mobile version of the Fulcrum platform, an example of which can be seen in Figure3.12, to create individual records for each of the surveyed points of interest during their deployment. They can capture images swiftly on the go in the application. These images are automatically geotagged and uploaded into the Fulcrum platform, where



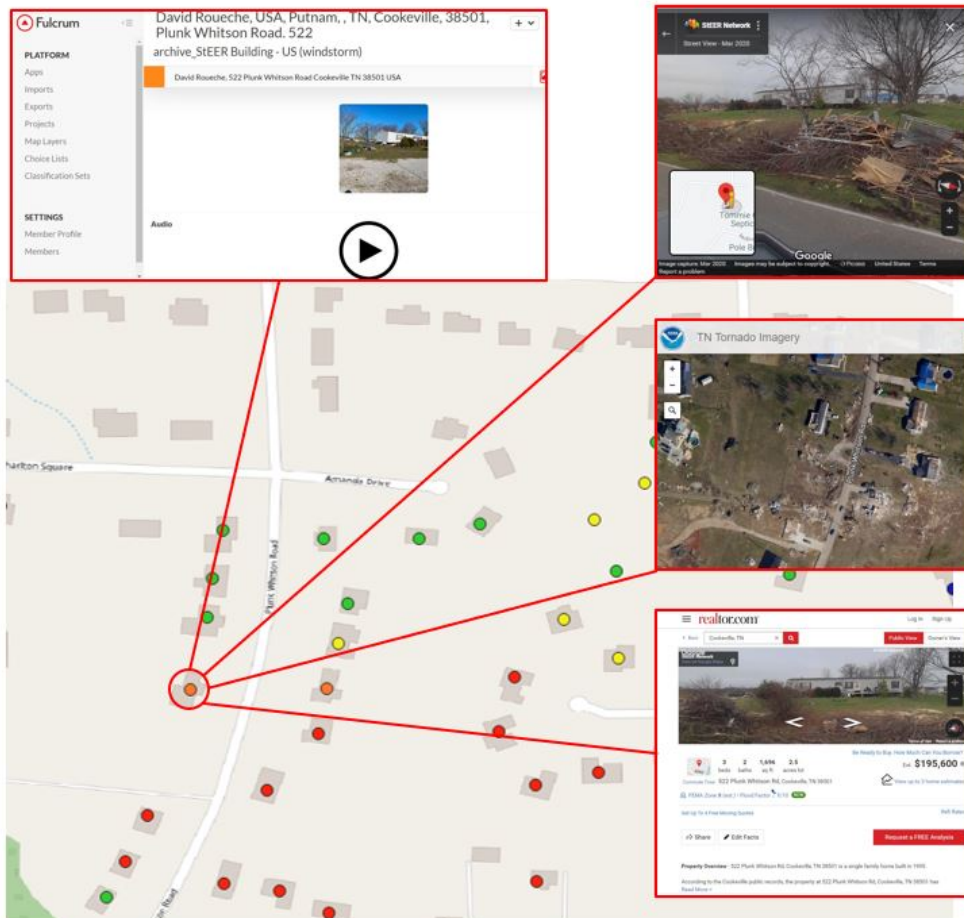


Figure 3.10: An example of a record from the Nashville Tornado that occurred in 2020. The map shows the location of the surveyed structure along with all the data sources available for this record which include photographs and voice recordings captured by the FAST, street-level panoramic imagery uploaded to Google Maps, NOAA aerial imagery, and realty website information which was provided by realtor.com for this record.

they are viewed by the Data Librarians later on during the data enhancement process. FAST members are also able to record any specific details relating to load paths, or key information that is not discernible from the captured images, as a voice recording or as a text entry.

Fulcrum’s table format shown in Figure 3.4 provides a convenient interface for filtering the data for specific characteristics that are of interest, for example, a specific natural hazard event or a specific type of structure or damage rating. This particular feature in the application is especially useful for quality control purposes. Other modes to view the data are available, such as map view, which is shown in Figure 3.13. This view mode allows for

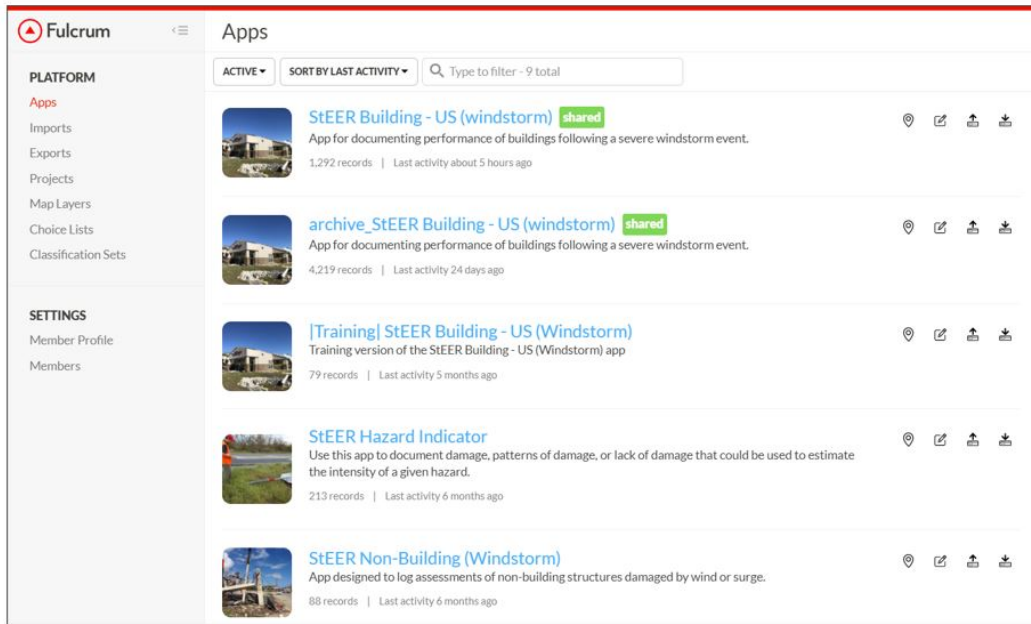


Figure 3.11: The Fulcrum platform landing page. This figure shows StEER’s windstorm applications which consist of StEER Building-US(windstorm), StEER Non-Building (Windstorm), and StEER Building-US (Windstorm). Also Visible is StEER’s training app used by the Data Librarians as well as the archive windstorm application.

viewing all records as points on the map. The points are color-coded based on the damage status of the record, and this view is similar to GIS software view. While this mode does not allow viewing all fields of the records as with the table format and does not allow the use of the filtering option, it gives a visual representation of the areas most damaged by the event and can bring attention to structures that had unique behavior. Such structures could serve as interesting case studies that can undergo further analysis. The third mode of viewing the data is a combination between the table mode and the map mode, shown in Figure 3.14, which combines the benefits of both options: it allows the use of the filtering feature while also displaying the points on the map.

Most edits are made directly within the Fulcrum web platform during the DE/QC process. The platform keeps track of every edit through version control, which includes when changes were made and who the changes were made by. The platform also logs editing time. In addition to the edits to individual records, edits can be made in bulk by importing



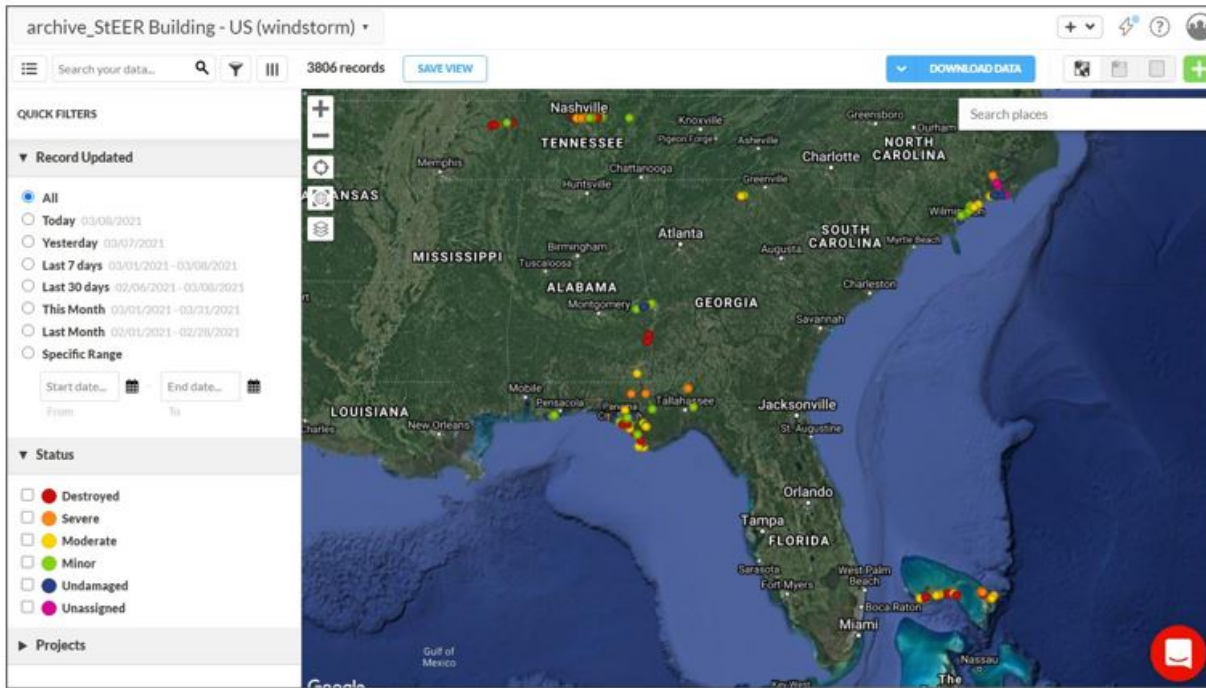


Figure 3.13: Map view in SteER’s archive application showing the records as points on the map. The colors represent the different damage states of the surveyed structures.

software. NOAA aerial imagery was accessed as a WMTS services and the assessment locations were overlaid on the imagery. A graduate student that is experienced with the data enhancement process was timed using the regular process that was described above and again using this new method for the Nashville tornado dataset. The roof structure damage, roof sheathing damage, and roof cover damage were the only fields tested because that is the data that can be extracted from the NOAA aerial imagery. The duration to the finish the damage assessments using this proposed method was approximately 50% less than the regular methodology. However, when the remaining steps of data enhancement were continued, it became necessary to reevaluate the roof damage assessments using the FAST ground-based photos or other sources to improve the accuracy of the GIS-based method. The error rates in the damage defined using aerial imagery only, though never precisely quantified, were high to the extent that roof damages had to be manually rechecked anyway using the FAST photos or street-level panoramas, which reduced the efficiency of this approach. Errors

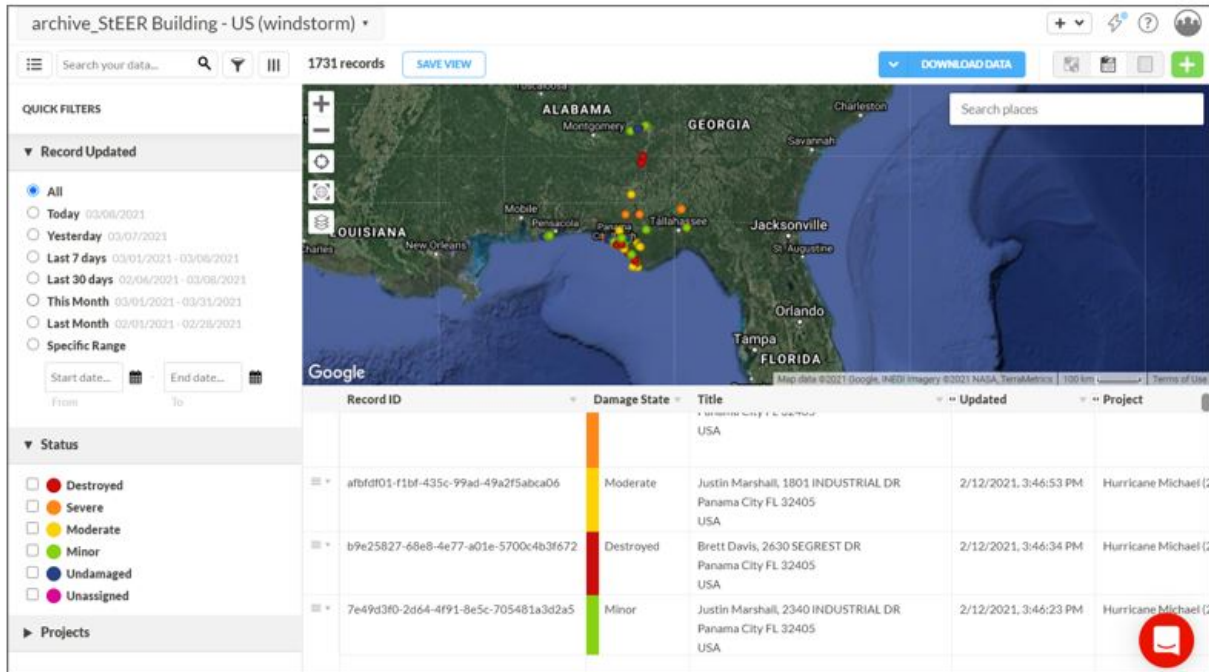


Figure 3.14: Map and table combination view in StEER’s archive windstorm application. This view allows the use of the filtering options in the table view while maintaining the map view at the top.

would include roof sheathing or roof cover damage not being visible in the NOAA imagery due to shadows or poor image resolution, when it was clearly visible in the high-resolution ground-based photos. While there may be other methods to improve the overall workflow, those performing the DE/QC process have all opted to utilize the record-by-record approach, working within the Fulcrum web platform, for most steps of the DE/QC process.

### 3.4 Post Processing Tasks and Considerations

The quality control process outlined in the framework targets the three distinct areas of data quality defined by Fox et al. (1994), which explained that data quality relates to three areas: the quality of the model that supports the data, the quality of the data values, and the quality of the representation of the data. The quality of the model that supports the data is tackled through revising the StEER’s DE/QC handbook used by the Data Librarians and clarifying unclear language or methods for extracting data based on feedback from the



Data Librarians. StEER's applications on the Fulcrum platform are also judiciously refined as needed by revising the options available for each field based on the needs encountered in the data enhancement process. The quality of the data values is addressed through the data quality dimensions defined by Fox et al. (1994), which are comprised of accuracy, currentness, completeness, and consistency. Finally, the quality of the representation of the data is addressed by developing clear guidelines in the handbook for each data entry in the datasets. The data also goes through rigorous quality control checks that include spot checks throughout the data enhancement process to make sure that the Data Librarians are consistent and to avoid duplicated errors. Other quality control checks that address the quality of data representation include automated checks that focus on the formatting of the data to ensure consistency.

The main portion of the quality control process in this framework, falls within the quality of the data values. The quality of data values is addressed by the data quality dimensions (accuracy, currentness, completeness, and consistency).

### **3.4.1 Accuracy**

To ensure a high level of accuracy, reliable data sources are used in data enhancement, which were discussed in detail in Section 3.3.3. Furthermore, the open-line communication with experienced supervisors in meetings or the Slack workspace provides a chance to correct common mistakes and avoid them in future assessments, which increases overall accuracy. Additionally, Data Librarians are asked to explain their methodology in data enhancement, their approach in damage assessments, and any assumptions that they made during the process.

Detailed quality control checks, which consist of manual and automated checks, play a key role in increasing the accuracy aspect of the quality of data. Manual checks are conducted on the Fulcrum platform in the designated application. The filter feature in Fulcrum is used heavily. First, each field in the dataset is checked for unreasonable or blank entries. This is

done by using the "Select Specific Values" Feature which shows a list of all the values entered into that field. Once an unreasonable entry is detected, the data is filtered to show those entries, so they can be manually checked. After the fields are checked for both unreasonable entries as well as blank entries, a process of multi filtering starts. Multi filtering refers to searching for fields that could be correlated with one another, which are then filtered and cross-checked. An example of this would be the First Floor Elevation field and the Understory (%of Building Footprint) field. Only an elevated structure would have an understory, so due to this, the Understory (%of Building Footprint) would be filtered for any value that is not blank and the First Floor Elevation field would be checked. If any record contained a blank entry in that field, it would require manual checking and reassessment of the First Floor Elevation. In a similar process, the opposite should be checked as well, any record with an entry in the First Floor Elevation field should have a value entered in the Understory (%of Building Footprint) field even if that value is zero. This method requires thinking of each of the fields and what other possible fields could be correlated with them. Table 3.6 shows examples of multi filtering checks performed in this framework and fields that are correlated with one another as well as a brief description as to why they are filtered together.

Moreover, any observations or intel from the FAST are used in further quality checks. For example, if a cluster was noticed to mostly have laminated asphalt roof shingles by the FAST during field reconnaissance, any value that differs from that is checked and the roof cover assessed once more. Another example would be if the FAST did not see any major damage on-site in a specific cluster, any records in that cluster that are marked as destroyed or severe in the damage status field would be checked as well. After this step is done, a list of tasks for the Data Librarians is formulated and any errors that were found are corrected and any blank entries are checked and if the data is available are filled in then the fields are checked again in the same way to make sure that no records were missed during the process.

Furthermore, a randomized validation is performed to evaluate the state of the data. A script written in Python selects a random sample between 7% -15% of the dataset for further

Table 3.6: Examples of fields that are multifiltered during quality control.

<b>Correlated Field 1</b>	<b>Correlated Field 2</b>	<b>Description</b>
Damage State	Overall Damage Ratings	The Damage State field represents the maximum of the three overall damage ratings.
EF Scale DI	Occupancy	DI in the EF Scale is based on the type of occupancy.
Understory (%Building Footprint)	First Floor Elevation	Only an elevated structure would have an understory
Occupancy	Number of Stories	A Single Family home would typically have 1-3 stories.
Foundation Type	First Floor Elevation	Elevated structures typically have piers for foundation.
Fenestration Protection	Damaged Windows(%)	If all windows were covered by protection, damage would not be visible.
Sectional /Rollup /Garage Door Present?	Sectional /Rollup /Garage Door Failure	If the structure does not have a garage door, the field indicating garage door failure should be blank.



quality control. The sample is equally divided by the number of Data Librarians and each of the Data Librarians will get a set of random records. When writing the automated script, it was taken into consideration that the main Data Librarian that worked on most of the data enhancement for the record does not receive that record in their set of random records and that is done to try to find any inconsistencies in assumptions or misunderstandings in any of the methods used in data enhancement that were not caught by this point. Each Data Librarian will reevaluate each of the records checking for any errors in the entries while taking note of the field with the error in an Excel Spreadsheet. Once the random sample is reevaluated, another Python script calculates the error percent for the entire random sample as well as the error percent for each of the fields in the data. This provides a quantitative measure of the accuracy of the dataset, and informs whether it needs further quality control and what specific fields are the weak points in the data. Any fields that have a significant error percent (generally 5% or higher) are flagged for re-evaluation. Once the weak areas of the data are addressed, a new random sample is generated and once it is reassessed by the Data Librarians, the updated error percents are calculated. When the error percents reach an acceptable limit (<5% is the target value), the final error rates calculated are reported in the data report associated with the publication of the dataset on DesignSafe-CI.

### **3.4.2 Currentness and Completeness**

The currentness aspect of data quality is addressed by taking into consideration the time that the data was captured by the data source. This is an important point to consider in post-windstorm reconnaissance because rescue, cleanup, and repairs will alter the state of the structure and will cause errors in damage assessments particularly. To avoid these types of errors, noting the date of the landfall of the windstorm and relating that time to the time data sources captured their data help in making sure the data collected reflects the post-windstorm state of the structure, and in prioritizing data sources for certain fields.

Completeness of the data can be measured by the percentage of filled fields in the data as well as referring back to the QC codes.

### **3.4.3 Consistency**

The consistency aspect of the quality of data is assessed by consistently treating unique cases and uncertainties in a similar manner each time. This is achieved by having clear guidelines in the DE/QC handbook, the guidance in training sessions, as well as the open communications with supervisors. Another aspect of consistency is that data needs to meet known guidelines. An example of this is checking the Wind Damage Rating field, which is assigned using component-level damage ratios. For this reason, the component-level damage ratios should be consistent with the assigned wind damage rating. To do this, a MATLAB script is used to find a calculated Wind Damage Rating value using the entered component level damage ratios and compares the result with the value that was entered by the Data Librarians in the Wind Damage Rating field. Once the records with inconsistencies are identified, they are checked again manually and corrected as needed.

### **3.5 Data Librarians**

The data enhancement process requires a detail-oriented approach, and it helps to have a background in construction or structural engineering to understand the terms being used, which makes engineering students an ideal choice for the role of Data Librarians. The use of students has the added benefit of exposing them to the real-world performance of structures, and training them further in the identification and assessment of common building systems. Thus, civil engineering students that are interested in the field of natural hazards engineering and the impacts of dynamic loads on structures were recruited, hired, and trained to perform the data enhancement process. A training regimen was developed that included a detailed handbook (Roueche, 2019), pre-recorded webinars, creation of a diverse set of training records (the same incomplete assessments that could be processed multiple times

by different students), dedicated workspace in Slack, and regular meetings for review and coordination.

Training Data Librarians starts by introducing the student candidates to the data enhancement and quality control handbook that serves as a comprehensive guide for the process. After the students are familiarized with the process through the handbook, a carefully selected set of training records is used to test their new knowledge. This provides a hands-on experience that is identical to the process performed on real records, thus allowing the candidates to quickly get familiar with the Fulcrum platform and the various data types that are used in the process, as well as identifying any areas of uncertainty. Over time, the use of the same training records by multiple trainees will also allow for statistics to be quantified on the most frequent mistakes. The candidates are given time to perform the data enhancement on the training records following the handbook. After this, they have an opportunity to get any questions answered in an online group training workshop. Based on the evaluation of accuracy of their assessments of the training records, the candidates either move on to the next step in the process as Data Librarians if they performed well, or will be retrained and have a chance to ask more questions if there were too many mistakes in their completion of the training records. After they perform adequately on the training records (an exact criteria for pass/fail has not yet been defined), the next step is to give the new Data Librarians small tasks in real datasets (e.g., verifying addresses, finding year built, classifying roof shape) and closely monitor their performance, providing them with feedback and answering any questions they may have. Once the Data Librarians gain confidence in the process and their performance is consistently satisfactory, they will start working more freely on the data enhancement process with periodic spot checks from a trained supervisor (an experienced graduate student or faculty member) that are familiar with the process.

For continuous learning and improvement, a Slack workspace channel (provided through the NHERI DesignSafe-CI), is used to provide a dedicated space for the the Data Librarians to ask questions, discuss uncertainties or difficult records, and share progress. An example

of this is shown in Figure 3.15. Other experienced Data Librarians or individuals that are supervising the process can share their findings and any tips or helpful resources. Weekly meetings are also held with the Data Librarians to further clarify any uncertainties as well as plan for future tasks to ensure continuous progress. The ease of communication using the Slack workspace and the weekly meetings together allow for a continuous feedback loop where areas of the process that need further clarification or require more examples are identified and used to update the data enhancement and quality control handbook. This provides consistency in the assumptions used in the assessments and sheds light on parts of the process that have more inherent uncertainty.

### **3.6 Case Study**

The DE/QC process is an iterative detailed process as can be seen from the discussion in the previous sections. To better understand the level of detail required, a case study showcasing the steps performed each time in the process along with real-life examples from StEER's datasets will be discussed in this section.

The data enhancement process starts from the windstorm applications on Fulcrum that house all points of interest surveyed by the FAST. The Data Librarian will select a record that has not yet been processed to start working on, based on priorities assigned by the supervisor. The DE/QC handbook is referred to for guidance on how to collect the necessary data for each of the fields during this process. Each record pertaining to a certain assessment stores all the standardized data collected from the different data sources during the data enhancement process. The first step is to make sure that the location and address of the records are correct. That is done using the coordinates that are available through the automatic geotagging feature in Fulcrum when a record is created. Using the photos of the structure, the coordinates, and supplemental data sources such as Google Maps or county GIS platforms/data, the location and address for the structure is confirmed.

steer\_deqc 8 | Add a topic 12 + i

---

**Christian Brown** 2:10 PM October 27th, 2020

Is the garage door rating on the new cluster known?

**David Roueche** 2:14 PM

Likely not wind-rated but no, we don't know for sure.

**David Roueche** 4:22 PM

The streetview imagery for both of the new streets is available now on Mapillary


---

October 28th, 2020

**Olivia Childress** 2:01 PM

Hi for this record would it be best to include the boards with the wall cladding as brick and wood boards?

Screen Shot 2020-10-28 at 1.57.24 PM.png



**Hadijah Rawajfih** 2:03 PM

Yes, we have been including all the types of the cladding present.

**David Roueche** 2:20 PM

The vertical oriented panels like that should probably be classified as a plywood siding, not wood boards. I know it's a little hard to see here, but I would either use the Plywood Siding wall cladding or Non-engineered wood panel wall substrate, with no cladding.

**Christian Brown** 4:57 PM

Is that roof damage or just a shadow or wear?u6

Figure 3.15: Example of Slack workspace communications.

The Data Librarians will refer to county tax websites and realty websites to extract building attributes which include year built, number of stories, roof cover material, and wall siding of the structure. The occupancy type of the structure can be extracted from these websites as well. While attributes like number of stories and cladding materials are often visible in the available imagery being reviewed, these websites along with imagery on Google

Street View of the structure pre-windstorm event are helpful to collect building inventory information on buildings that were completely destroyed and have no post-windstorm evidence left to analyze.

Next, the photos taken by the FAST are examined carefully and analyzed to extract the information that is needed to populate the remaining fields in the record. It is often necessary to anchor what side of the structure the Data Librarian is looking at in each photo by looking at the surroundings of the structure, for example, by looking at the location of a pool as in Figure 3.16 or a unique feature of the structure as shown in Figure 3.17. Figure 3.16a shows that a pool is present in the back left side of the structure which could act as an anchor to that side. When looking at Figure 3.16b, the pool indicates that the sides visible are the back and left sides of the structure due to the location of the pool in this photograph. Figure 3.17a shows that this structure has an outdoor staircase located on the right side of the building. When looking at Figure 3.17b, it can be seen that a portion of the staircase is visible where the red arrow is pointing, and that leads to the conclusion that the sides visible in this photograph are the right and back sides of the structure due to the location of the staircase. From these two examples, it can be seen that anchoring the sides in this method will help in visualizing the different sides of the structure with respect to one another and leads to a more accurate damage assessment. The audio recordings are then listened to and the information in them is transcribed and entered into the designated fields.

Street-level panoramic imagery is also examined to get a better perspective of the structure and check any sides that were not visible in the photos taken by the FAST. Street-level panoramic imagery helps orient the different sides of the structure with respect to one another and gives a different angle from which the structure can be seen. An example of this is shown in Figure 3.18a, which is an image taken in D2D and the roof is not visible, but the street-level panoramic imagery shown in Figure 3.18b is taken at a higher elevation due to the location of the road where the vehicle that the equipment is mounted on is driving, which gives a better view of the roof. During this process, the Data Librarians also rely



Figure 3.16: An example of how to anchor the different sides of the structure based on details around the structure. a) shows a pool at the back left side of a structure impacted by Hurricane Michael (2018). b) shows a second image of the structure. The location of the pool relative to the structure indicates that this is the back of the structure. The arrow points to the pool.



Figure 3.17: Another example on anchoring the sides of the structure based on details in structure or the surroundings. a) the left side of the structure, which was impacted by Hurricane Michael (2018), has an outdoor staircase which is indicated by the red square. b) a small part of the staircase can be seen where the red arrow is pointing. This indicates that this is of the back of the structure.

on essential observations and notes made by the FAST on details that need more expertise, such as connection types and information about load paths, to fill in the more detailed fields such as wall anchorage type and roof-to-to-wall attachment fields.



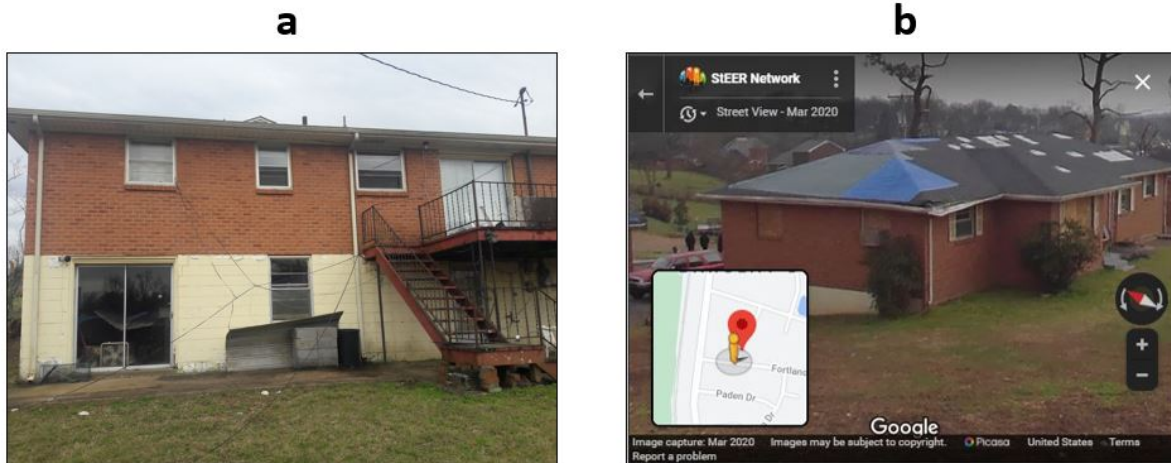


Figure 3.18: Two images of a structure impacted by the Nashville Tornado in 2020 that show the benefit of using street-level panoramic imagery in addition to D2D photographs. a) A photograph taken in D2D assessments where it can be seen that the roof is not visible in this photograph. b) A screenshot of the street-level panoramic imagery uploaded to Google Maps of the same structure which was captured by the FAST and it shows that the roof is more visible in this image.

At this point, an idea of the extent of the damage to the roof is formed, but to confirm it, orthogonal or aerial imagery is used. Satellite imagery such as NOAA aeriols, point clouds if they were collected, or Pictometry imagery if available, are used. This different angle provides a more accurate damage assessment. It is worth noting that access to more than one data source is also very helpful for damage assessments, because there might be an obstruction like clouds present in NOAA aerial imagery that limit or block the visibility of roofs. Some areas could also be out of the range covered by NOAA's aerial imagery or Pictometry imagery. Additionally, if the roof is damaged, this type of imagery can help identifying the roof system used in the structure. Point clouds and Pictometry imagery can be used to get better precision in fields that require measurements such as roof slope, first-floor elevation, and overhang length. If this imagery is not available, the Data Librarians will rely on using elements of the structure with common or standard dimensions such as standard door height in the US or the dimensions of wood planks used in timber construction. The data sources that consist of oblique imagery are also used to view the sides of the structure that were either not clear in the images taken by the FAST or to view the sides of the



structure that were not captured in photos due to accessibility. When each component of the structure is assessed and assigned an estimated damage ratio, an overall damage assessment is specified.

### 3.7 StEER Data Archive

Since StEER’s formation in 2018, it has responded to 12 windstorms. A list of these events can be seen in Table 3.7, where the event name, year it occurred, as well as the type of response StEER provided are shown. For more major events that resulted in widespread damage, such as Hurricane Michael, FASTs were deployed and field assessments were performed. Other storms that had less impact such as Hurricane Zeta and Hurricane Eta, StEER’s response was limited to an event briefing, which focused on major lessons learned for policy and practice.

Table 3.7: A list of windstorm events that StEER responded to since its formation in 2018 as well as the type of response that StEER provided for each of the events. \*P-VAT refers to Preliminary Virtual Assessment Report.

<b>Event Name</b>	<b>Reponse from StEER</b>
Hurricane Florence (2018)	FAST and EARR
Hurricane Michael (2018)	FAST,P-VAT*, EARR, Dataset
19 January 2019 Tornadoes in the Southeastern US	FAST and EARR
3 March 2019 Tornadoes in Southeastern US	FAST and EARR
14 March and 25 April, 2019 Cyclones Idai and Kenneth in Mozambique	Event Briefing

22 May 2019 Jefferson City, MO Tornado	EARR
28 May 2019 Linwood, KS EF4 Tornado	EARR, Dataset
Hurricane Barry (2019)	Event Briefing
Hurricane Dorian (2019)	PVRR, EARR, and FAST
Typhoon Hagibis (2019)	Event Briefing
10.20.2019 Dallas, TX EF3 Tornado	Event Briefing, FAST
3 March 2020 Nashville Tornadoes	FAST, VAST, PVRR, and EARR
Hurricane Laura (2020)	PVRR-EARR, FAST
Hurricane Sally (2020)	Event Briefing and FAST
Hurricane Delta (2020)	Event Briefing
Hurricane Zeta (2020)	Event Briefing
Hurricane Eta (2020)	Event Briefing
2020 Midwest Derecho	FAST
2021 Fultondale, AL Tornado	FAST

Up to this date, StEER has collected 5492 records in total that span over 11 windstorms. Figure 3.1 shows the distribution of records per event by the coded QC stage. These records are in different progress stages due to the time and effort it takes to process the data and perform data enhancement and quality control. Based on the QC codes detailed in Section 3.6, Figure 3.19 shows the different stages the 5492 records are at and the percentage of records at each stage. Nearly 35% of the records are still in QC Stage 0 or 1, indicating minimal processing has been completed. This highlights the need for increased efficiency in

the data enhancement and quality control process so that datasets can be completed and published sooner. A step in that direction is developing the automation framework discussed in Chapter 4.

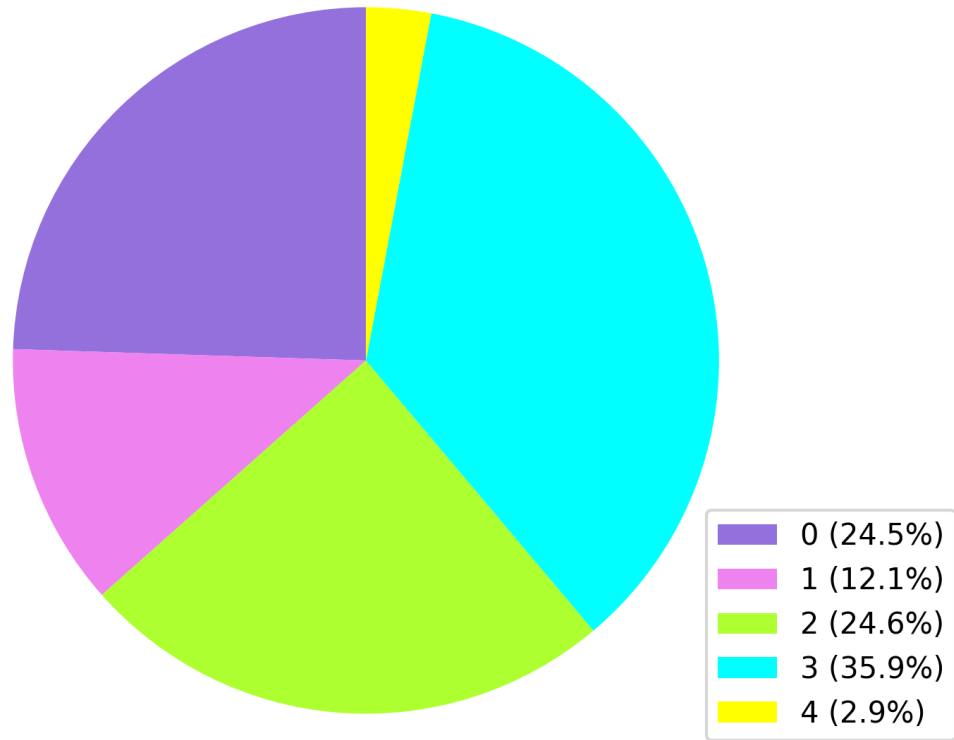


Figure 3.19: StEER’s data distribution into QC codes. QC codes that refer to records that might have more uncertainty than other records, which are denoted with the letter ”e”, are grouped into their equivalent number QC code in this plot. for example, 2e was included in 2.

### 3.8 Chapter Summary

This chapter discussed the development of a protocol for post-windstorm data enhancement and quality control. It covers the types of data collected by the FAST and highlights the inconsistencies and gaps in such data. The various steps of the DE/QC process developed to address these shortcomings are described in detail. The chapter covers the three main steps of the developed DE/QC process which consist of pre-processing, main processing, and post-processing of the FAST collected data. The discussion of pre-processing details locations

validation, duplicate removal, and address verification. The main processing discussion covers extracting building and structural attributes as well as performing damage assessments. Finally, the post-processing discussion focuses on data quality dimensions which consist of accuracy, currentness, completeness, and consistency.

This chapter also details the role of Data Librarians and the development of a training regimen that prepares them to perform the developed DE/QC process. Finally, a case study that walks through DE/QC and provides real-life examples is given. The developed DE/QC process aims at producing high-quality datasets, but the process can take months to complete. Thus, it is worthwhile to explore ways to automate parts of the process, which will be the focus of the next chapter.

## Chapter 4

### Automation Framework

As shown in Chapter 3, the data enhancement process described in Section 3.3 is the most time-consuming step of creating post-windstorm reconnaissance datasets. Thus, efforts to make this process more efficient can benefit researchers and ultimately society by making the datasets available for analysis sooner. One way to speed up the process is by automating certain aspects of data enhancement that do not rely on engineering assessments and knowledge. To aid in accelerating key processes in post-windstorm reconnaissance data enhancement, a preliminary framework that automates various aspects of the data enhancement process was developed for this thesis using Python, building on existing open-sourced components that have been developed by the NHERI SimCenter.

#### 4.1 Framework

Figure 4.1 shows a flowchart of the framework process workflow that was designed for this work. The framework was developed to demonstrate how various technologies such as web scraping and machine learning can be utilized for automation. The framework is split into dedicated modules that can be implemented as a whole, or separate modules can be integrated into the manual workflow to speed it up. It can also be used as a first-run that is then enhanced through manual and automated quality control checks. As an example, the framework includes a reverse geocoding module that can be used to obtain addresses for a set of provided coordinates, which is otherwise a time-consuming process if done manually. The addresses could still be validated later using other methods such as a spatial join with county tax data. Other modules take advantage of web scraping, along with using BRAILS and SURF, to fill in various building attributes such as year built, number of stories, roof

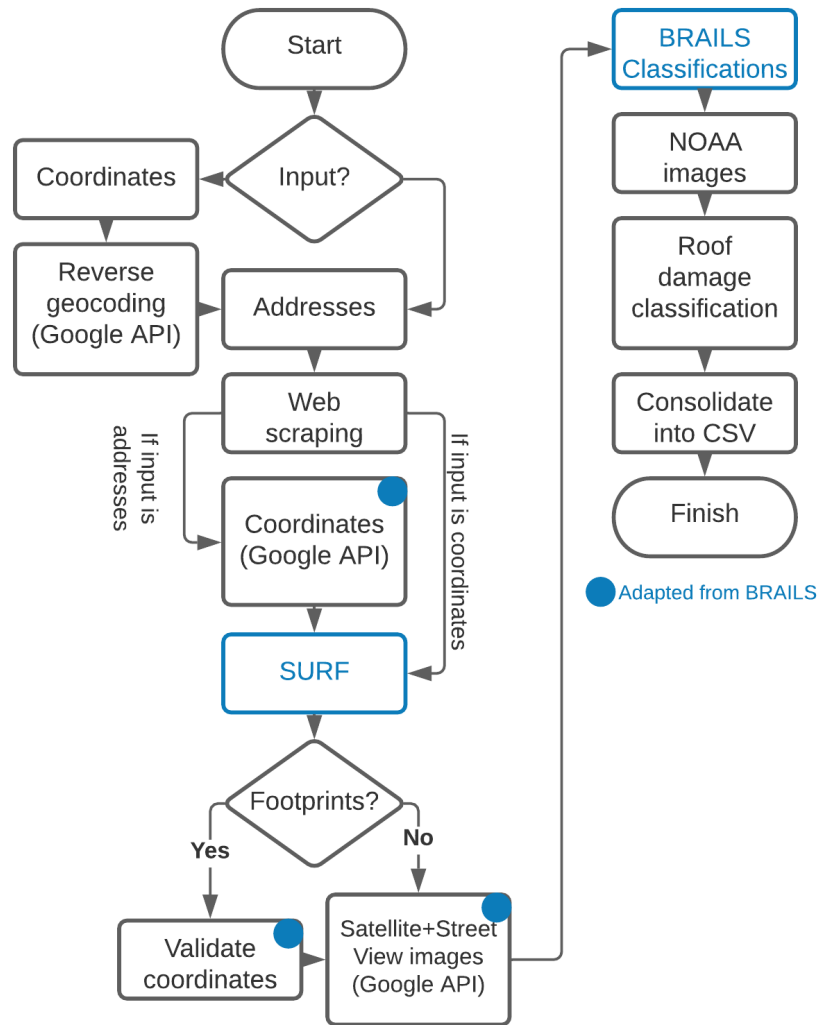


Figure 4.1: Automation framework flowchart.

type, and many more. To ensure a higher level of accuracy, the Data Librarians can then use other sources including imagery collected by the FAST for verification and quality control of the framework results. A damage classification module that uses machine learning to predict roof damage is also included and can classify the roof damage for individual buildings into one of five classes: undamaged, minor, moderate, severe, or destroyed.

To start the framework, the user needs to provide a Comma Separated Values (CSV) file representing a list of buildings of interest as an input. The user has two input options: a list of coordinates, or a list of addresses. A list of coordinates would be the input option

of choice if field reconnaissance was performed, as the surveyors obtain precise coordinates of the structures of interest from the field. The list of coordinates input option could also be used if virtual reconnaissance was performed, where points of interest are selected using GIS software and satellite imagery of the impacted area. Conversely, a list of addresses of interest can also be used as input instead. If coordinates are provided, a reverse geocoding module utilizes Google's Application Programming Interface (API) to obtain the addresses associated with the input coordinates. Next, the addresses that were obtained through the reverse geocoding module, or given as input, are passed to the web scraping module. The Web scraping module is used to find building attributes corresponding to the structures located at the addresses of interest, such as year built, number of stories, occupancy type, roof type, and foundation type. After that, an image downloading module again utilizes Google's API to download Google Street View and Google satellite imagery for each address using the provided coordinates. If addresses were provided as input instead of coordinates, this module also uses Google's API and geocoding to get the coordinates for each of the addresses. The downloaded images are then passed to the BRAILS module, which uses pretrained BRAILS machine learning models to predict building attributes of the structures of interest, including roof type, number of stories, and year built. The building attributes found from the BRAILS module are used to validate the results from the web scraping module, as well as fill in any attributes where the web scraping module did not yield any results. To further enhance the data, a module that incorporates SURF is implemented to fill in the gaps present in the collected data after both the web scraping and BRAILS modules are used, based on spatial relationships. Next, another image downloading module is used to download NOAA post-windstorm aerial imagery of the individual roofs of structures using the provided or geocoded coordinates. These images are then passed to a roof damage prediction module that incorporates a machine learning model, trained on NOAA aerial imagery specifically for this framework, to classify the impacted roofs into the five damage categories listed above. Finally, all the data produced by each module, which consists of

addresses, coordinates, building attributes, and roof damage classifications, are consolidated into a single CSV file.

The desired output from the framework is a dataset that could be ingested into the standardized post-windstorm reconnaissance datasets generated by StEER. For this reason, an input option for coordinates collected in the field was added. While the coordinates are used in SURF, BRAILS classifiers, and for capturing images from NOAA aerial imagery, the addresses are still needed for web scraping building attributes. The coordinates that are either collected in field reconnaissance or selected in StEER’s Rapid Survey response strategy are provided as inputs for the reverse geocoding module to get the addresses associated with the coordinates. The reverse geocoding module uses custom code to extract the addresses present in the GeoJSON files retrieved from the Google API. The extracted addresses are then reformatted to follow the StEER data format: Address sub\_thoroughfare (street number), Address thoroughfare (street name), Address locality (city), Address sub\_admin\_area (county), Address admin\_area (state), Address postal\_code, and Address country.

## **4.2 Automation Techniques and Tools**

The explored technologies in the framework include web scraping as well as existing open-source Python packages including BRAILS and SURF, which will be discussed in detail. Additionally, multiple machine learning models are explored to find the best performing model to classify roof damage into five categories.

### **4.2.1 Web scraping Process**

Web scraping is the process of collecting data from websites, parsing it, and outputting the data into an easy-to-use format such as a spreadsheet. This process can be automated using published tools such as the Selenium Python package (selenium, 2018), which is an open-source automation tool that uses a web browser to access websites, capture their data, and process it. A Python module was written that uses Selenium to search for and extract



desired data using a Google Chrome WebDriver. The WebDriver is a tool that opens a Google Chrome window and allows Selenium to control it. The data is extracted by performing a Google search on each of the addresses provided by the user or obtained from coordinates through reverse geocoding. Depending on the attribute to be web scraped, a search term is automatically added to each address. Selenium is programmed to save the resulting Google search snippets, which are then parsed, and the attributes of interest are saved. In its current version, the web scraping module can support the following attributes: year built of the structure, number of stories, foundation type, occupancy type, and roof shape. The web scraping module is customizable, allowing the user to call the function for the desired attributes of interest only, instead of the entire list of attributes.

#### **4.2.2 BRAILS Overview and Incorporation in the Automation Framework**

BRAILS (Yu et al., 2019a) was introduced in Section 2.4.3 but will be described in more detail related to its integration into the automation framework. BRAILS is an open-source tool, developed by NHERI-SimCenter, that employs the use of machine learning and deep learning to create a building inventory on a regional scale by extracting the information from Google Street View and Google satellite imagery. BRAILS includes pretrained machine learning classification models, known as classifiers, that predict various building attributes. BRAILS also includes tools that can collect building inventory data for a specified region of interest and obtain all the addresses and coordinates for the buildings in that specified region. Because the framework developed for this thesis is focused on post-windstorm reconnaissance data and its enhancement, only specific coordinates and addresses of a representative sample from the impacted area are of interest. Therefore, BRAILS is incorporated into this framework to predict building attributes for these points of interest only, instead of entire regions.

As of the date of writing this thesis, BRAILS consists of six unique classifiers for identifying building attributes. A roof shape classifier, which classifies the roof into three distinct

classes: hip, gable, and flat. An occupancy classifier, which classifies the occupancy type into single-family and multi-family. A soft-story classifier, which identifies if the structure is soft-story. A year-built classifier, which classifies the year built into one of six different decades. A raised foundation classifier which identifies if the structure is elevated or not. And number of floors detector which predicts the number of stories the structure consists of. BRAILS also provides a general image classifier that enables the user to train the classifier on their own user-provided images. BRAILS classifiers use a CNN model, which falls under supervised learning, to predict building attributes. When BRAILS trained their classifiers, they used building attributes extracted from OpenStreetMap as labels for the training datasets. The images that are used for training and predicting are captured from Google Street View imagery and Google satellite imagery using the coordinates obtained from a geocoding step which is part of BRAILS' framework. Once the coordinates are obtained, Google's API is used to download the images.

The automation framework was inspired by BRAILS and parts of BRAILS' source code were adapted and incorporated into the automation framework. The parts of the automation framework that used adapted source code from BRAILS are indicated in the flowchart in Figure 4.1 and are marked with a blue circle.

The next step in the automation process is validating the coordinates using building footprints. This is an optional step that was also adapted from the BRAILS source code. If the user has building footprints of the area that contains the coordinates, it can be used to ensure that the coordinates of each building are within a building footprint. This helps ensure the images that are download from Google Street View imagery or Google satellite imagery or NOAA aerial imagery are accurate and represent the structures of interest. This step flags the coordinates that do not fall within a building footprint for further manual investigation. BRAILS suggests using Microsoft's AI-generated building footprints which consist of separate files for each state in the United States. The framework provides an

option to use one of these files or similar files as well. As mentioned previously, this validation process is optional and can be skipped by the user.

After the coordinates are validated, adapted source code from BRAILS is used in a module that downloads individual Google Maps and Google Street View images of each structure using Google’s API. Once the images are collected, they are passed to another module that uses BRAILS classifiers to predict building attributes. At this point of the automation framework, the dataset consists of coordinates, addresses, building attributes obtained from web scraping that include: year built, number of stories, foundation type, occupancy type, and roof shape, as well as building attributes predicted from BRAILS classifiers which include: roof shape, occupancy, year built, raised foundation, and number of floors. The overlapping building attributes from web scraping and BRAILS are both saved in the final output file for comparison.

### **4.2.3 Data Enhancement Using SURF**

The next step in the framework is to use the coordinates that were either provided as inputs by the user or obtained using the geocoding step, in a data enhancement module that uses SURF. SURF is another open-source tool developed by NHERI-SimCenter (Wang et al., 2021) that uses spatial uncertainty analysis on the data provided, in which the SURF model is first trained on a set of geocoded data labelled with a certain attribute of interest, where the SURF machine learning model learns the spacial patterns in the provided dataset. Then, with a user-selected number of nearest neighbors, the model can be used to predict that same attribute the model was trained on for geocoded points that are unlabeled. For a certain training data point, SURF requires the coordinates of that point and one label. The label has to be a number. Thus, if the user needs to use SURF to predict text labels for example, they would need to code them into numbers first. An example on how SURF can be used is with the web scraped number of stories. A SURF model can be trained on the data that was found, so for each record, the coordinates and the web scraped number

of stories label are given as inputs to train the SURF model. Then, the coordinates for the records that web scraping failed to find a number of stories for are used with the trained SURF model to predict their number of stories. It is important to point out that for SURF to work, the training data must include at least two different labels, otherwise there would not be any spatial patterns to learn.

Hence, the inclusion of the SURF module into the framework adds another layer of robustness to the attributes collected and maximizes the number of attributes that the framework can provide.

#### **4.2.4 NOAA Aerial Imagery in the Automation Framework**

Sections 4.2.2 and 4.2.1 introduced methods for automating the identification and classification of building attributes. For damage detection and classification, post-storm imagery is needed. NOAA's Remote Sensing Division collects NOAA aerial imagery and typically makes it publicly available a few days after a windstorm event. The imagery is described to have a ground sample distance that ranges from 15 cm to 50 cm per pixel depending on the event. NOAA aerial imagery was chosen for roof damage state detection because of the availability of the imagery, the wide-area covered by this imagery, as well as the number of storms captured by the team. However, in contrast to the process of downloading Google Maps and Google Street View imagery which is done through exploiting the Goggle API, no API exists for NOAA. Therefore, a different approach was needed to automate the NOAA imagery capturing process. For this, the Selenium python package was used again in a module that incorporates the provided coordinated into NOAA URL links. The URL links containing the coordinates are then used with Selenium and the Chrome webdriver to open a Chrome window that displays the NOAA imagery of the structure matching the coordinates. The module then saves a snapshot of the browser window. Due to this, the browser appears in the captured images and sometimes parts of roofs of neighboring structures appear as well. For this reason, code was added to the module to crop images after

they are captured, leaving only the roof of the structure of interest visible, and discarding the browser window and other unwanted artifacts. The uncropped screenshots are deleted automatically after they are cropped for convenience and to reduce confusion when the user looks at the framework outputs.

### 4.3 Classifying Roof Damage Using Machine Learning Algorithms

As discussed in Section 2.4.2, damage detection and classification using machine learning has been demonstrated by researchers in the community. The developments in this field constitute major strides towards a future where post-disaster reconnaissance can be converted to a fully automated process. This framework demonstrates small steps in that direction by combining several modern techniques to achieve a form of automation.

Alzubi et al. (2018) outlined six components that training a machine learning model is composed of. These components are data collection and preparation, feature selection, algorithm selection, model and parameters selection, training, and performance evaluation. For the framework, data collection is achieved by automatically collecting images of individual roofs from NOAA aerial imagery which was discussed in Section 4.2.4. Additionally, the coordinates to obtain images of the individual roofs were taken from StEER's database of post-windstorm data. Using points of interest from StEER's data enabled the use of StEER's damage assessment as labels.

A big focus in StEER's datasets and post-windstorm reconnaissance data, in general, is damage assessments, which are conveyed in StEER's data by component level damage ratios as well as a damage rating to quantify the overall damage condition. For this reason, a damage assessment aspect is needed in the automation framework. While roofs do not reflect the damage state of a structure precisely, they can give a general idea of the damage state of the structure. The automation framework uses NOAA aerial imagery to predict the damage states of individual roofs. Damage state criteria were adapted from the one used for the Wind Damage Rating field used by StEER, which can be found in StEER's

VAST Handbook: DE/QC - US Windstorms and is shown in Figure 3.6 for convenience. The criteria used for the roof damage states are the criteria in Figure 3.6 which relate to roof damage and include roof component damage ratios and are summarized in Table 4.1.

The roof damage state is intended to reflect the level of damage the roof sustained without taking into consideration that damage to other components of the structure may be present. Based on the guidelines for the Wind Damage Rating in StEER's VAST Handbook: DE/QC - US Windstorms, a structure could be assessed as minor damage if the wall cladding consisted of a damage ratio larger than 0% and less or equal to 15% and no roof damage was present. A structure like that would appear as an undamaged roof in aerial imagery but would be assessed in StEER's datasets as minor damage state. To avoid issues of this nature, records from StEER's database on Fulcrum were used, and the Wind Damage Ratings for the records were taken as labels for the training dataset. For each roof damage state considered, a group of records was selected and the coordinates, addresses, and Wind Damage Ratings for the selected records were taken. In this process, the Fulcrum records were filtered based on Wind Damage Rating and then NOAA aerial imagery of the individual roofs was automatically captured using the corresponding module in the automation framework. The images were then manually inspected to make sure that each roof in the training dataset sustained damage that aligns with the roof damage state criteria.

Another machine learning component is feature selection. Feature selection refers to selecting features in the data that enable increasing the efficiency of the machine learning model. Feature selection was left for the model to learn during the training in the current version of the framework. In the future, feature selection methods such as annotating the roof or the damage the roof sustained could be used to advance this framework further.

The model and parameters selection component was addressed by identifying the type of problem at hand. First, a flowchart in the documentation of the Python machine learning library scikit-learn, developed by Pedregosa et al. (2011), shown in Figure 4.2, was used as a rough guide to identify what type of machine learning algorithm is the most appropriate

for the task at hand. Following the arrows in the chart, the available data is more than fifty samples, the end goal is to predict a category (roof damage state), and the data at hand is labeled, from that, we conclude that this is a classification problem. Once the type of problem at hand is identified and knowing that classification is a form of supervised learning, our machine learning problem is defined as a supervised learning classification problem.

Table 4.1: Roof damage status criteria used in training the machine learning algorithms. These criteria are taken from Figure 3.6

<b>Damage State</b>	<b>Roof Cover Damage (%)</b>	<b>Roof Substrate Damage (%)</b>	<b>Roof Structure Damage (%)</b>
<b>No damage (0)</b>	0%	0%	0%
<b>Minor damage (1)</b>	>0% and ≤ 15%	0%	0%
<b>Moderate damage (2)</b>	>15% and ≤ 50%	1-3 panels	0%
<b>Severe damage (3)</b>	>50%	>3 and ≤ 25%	≤ 15%
<b>Destroyed damage (4)</b>	>50%	>25%	>15%

Saravanan and Sujatha (2018) explained that supervised machine learning algorithms use labeled data as examples to find an inferred function that can map new data to predefined labels. The data is divided into training data and testing data. The algorithm will make comparisons between the predictions it made and the correct labels when the model is being trained. Then the trained model uses what it learned from the training data on the testing data for evaluation of the performance of the model. Each classifier has defined hyper-parameters that can be tuned to optimize the performance of the model. scikit-learn is one of the most popular python machine learning libraries. It provides a wide collection of

machine learning models to choose from. It also has tools that help with the process of tuning the hyper-parameters of each model, such as GridSearchCV and RandomizedSearchCV, both of which require the user to provide a list of hyper-parameter values and these tools will find the optimal combination of the provided parameters. The specific algorithms that were selected and tested are discussed in Section 4.3.1.

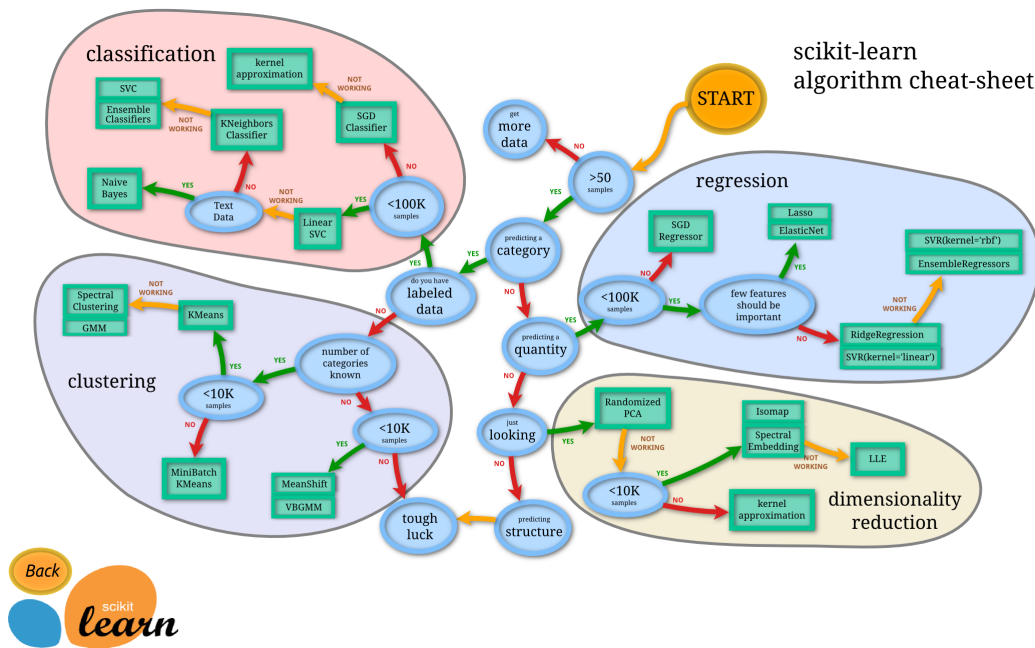


Figure 4.2: Flowchart taken from scikit-learn library documentation that gives guidance on how to choose the appropriate machine algorithm. It also explains in easy terms how to identify what type of machine learning problem is at hand

For training, two training datasets were created due to changes in NOAA aerial imagery resolution for one of the events used. When the automation framework was in the process of being developed, NOAA aerial imagery of the Nashville Tornado and Hurricane Laura was selected to be used to create a training dataset. The imagery for both events was available in zoom level 20, which is described to have a ground sample distance of approximately 15 cm per pixel. The training dataset images were collected when the available zoom level was still 20. Since then, the zoom level for Nashville Tornado imagery has decreased to zoom level 18 which is described to have a ground sample distance of approximately 50 cm per pixel. Due to the decrease in zoom level and to be able to run the framework on points that exist



in StEER’s data for comparison, an additional training dataset was created to train a new model for this zoom level. When the zoom level changed for NOAA aerial imagery from zoom level 20 to zoom level 18, the automatically collected images no longer represented individual roofs and instead, the images contained roofs of multiple structures. This caused the model that was trained on the images from zoom level 20 to perform poorly. The imagery that is at zoom level 18 needed to be cropped to a much smaller size that could capture individual roofs separately. The new size of the images after cropping is square images of size 50 by 50 pixels. The size of the images in the training dataset needs to be constant because each image is preprocessed by converting the image to an array of floating-point values of red, green, and blue, so the size of the array depends on the number of pixels that the image consists of. Thus, if the number of pixels in an image changes, a new model needs to be trained.

Thomas et al. (2014) and Yeum Chul et al. (2019) noted that when a training dataset is not equally distributed across the different classes, the trained model can predict with bias for the majority class. For this reason, the training datasets were selected with equal number of images per roof damage state (class). For NOAA, aerial imagery of zoom level 20, the training dataset consisted of 300 images with 60 labeled images per each of the five roof damage states. For NOAA aerial imagery of zoom level 18, the training dataset consisted of 285 images with 57 labeled images per each of the five roof damage states.

The performance evaluation component was tackled by testing the trained model using 20% of the training datasets described. The performance of the trained model was determined by using the F1 score with 5 fold cross validation, which is described as the weighted average of precision and recall. The confusion matrices were plotted for each of the models for evaluation as well. Performance evaluation will be covered in more details in the following sections.

### 4.3.1 Exploring Different Algorithms and Approaches

The previous section demonstrates how roof damage state classification is a supervised learning classification problem. With the machine learning problem clearly identified, the next step is to choose an appropriate model to be trained. Many machine learning libraries provide a diverse set of models that are easy to train and optimize for a specific task. The previously discussed scikit-learn Python library was chosen for this framework due to its powerful, streamlined, diverse, and easy-to-use set of models. Specifically, three scikit-learn models were tested to perform roof damage state classification. The three models, also known as classifiers, are Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), and Multi-layer Perceptron (MLP). Each of these models was trained on the two previously mentioned datasets from NOAA aerial imagery with different resolutions, and their performance was evaluated using accuracy and the F1 score. F1 score refers to the weighted average of the precision and recall and can be computed using the following equation:

$$F1 = \frac{2 \times \textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$

Recall and precision are calculated using the equations below, where TP refers to true positive, FN refers to false negative, and FP refers to false positive:

$$\textit{Recall} = \frac{TP}{TP + FN}$$

$$\textit{Precision} = \frac{TP}{TP + FP}$$

According to the documentation, the closer the F1 score is to one, the better the performance of the trained model. Additionally, accuracy is a measure of the percentage of predictions that exactly match the provided labels. Similarly to the F1 score, the closer the accuracy value is to one the more satisfactory the performance of the trained model is. Additionally, confusion matrices were constructed to visualize the performance of each trained model. A

confusion matrix compares the predicted values to the true values and shows the correct as well as the wrong predictions for each class.

Below is a detailed discussion of each of the three models, including a brief description of the models, how they were implemented, and their performance evaluation.

### 4.3.2 Stochastic Gradient Descent (SGD) Classifier

#### i. Overview and Method

The first classifier that was tested was the scikit-learn `SGDClassifier`. As per the documentation of this classifier (scikit learn, 2020b), SGD is used for supervised learning classification and uses stochastic gradient descent learning on a regularized linear model. Stochastic gradient descent refers to an algorithm that iteratively fits the training data until a chosen loss function is minimized. At each iteration, the fitting error for a random data point is calculated using the selected loss function. The gradient of the loss function is then taken with respect to the fitting parameters (intercept and slope) at that point, and, along with a specific step size for a selected learning rate (e.g.  $\text{step size} = \text{fitted parameter} \times \text{learning rate}$ ), updated fitting parameters are calculated. The process is repeated to travel down the slope of the loss function until the loss function is minimized or the user-specified maximum number of iterations is reached. The process is called stochastic because it does not compute the gradient using all data points, instead, it uses a random data point at each iteration, which significantly reduces the processing time needed. Some of the major hyperparameters that influence this classifier are the learning rate of the function, which controls the step size taken in each iteration, the selected loss function, and the maximum number of iterations. This classifier allows the use of several loss functions and penalties. Regularized refers to the penalties that are imposed on the loss function as a result of overfitting the data. As with the majority of supervised classifiers, two arrays are required to train an SGD classifier. One array for the training samples and another array for the labels associated with the training samples. For multi-class classification, which is the case for roof damage state classification,

the classifier uses a one-versus-all strategy. For each unique class, a binary classifier learns to differentiate between that class and all remaining classes using a decision boundary. The results are then combined into one final model that is used to predict a new set of data.

## ii. Roof Damage Classification Using SGD Classifier

For roof damage state classification, the inputs are two arrays, an array  $X$ , consisting of all images, where each image is converted to an array of floating-point values that represent the red, green, and blue percentages of each pixel in the image, and an array  $y$ , which consists of the roof damage state classification labels associated with each image. The `SGDClassifier` is sensitive to feature scaling, so the  $X$  array is rescaled using the standard scaler function from `scikit-learn` to avoid poor training results. The standard scaler standardizes each input so that the mean is equal to zero and the variance is equal to one. Next, the training dataset (The  $X$  and  $y$  arrays) is split into two portions where 80% of the dataset is used for training ( $X_{train}$  and  $y_{train}$ ), and 20% is used for testing ( $X_{test}$  and  $y_{test}$ ). This can be performed by using yet another useful `scikit-learn` tool: the `train_test_split` function. Other than splitting the dataset in two, the `train_test_split` function also has options to shuffle the data in each dataset, as well as to stratify the split datasets. Both `shuffle` and `stratify` options were enabled when `SGDClassifier` was used, as it requires the data to be shuffled beforehand. Stratification is generally good practice when performing train-test splits as it ensures that both the training and testing datasets are not biased towards a specific class.

The next step is to train the `SGDClassifier` on the training dataset, where the model tries to fit the  $X_{train}$  data according to the  $y_{train}$  labels, under the constraints of the hyperparameters discussed above. To find the optimal hyperparameters for the best `SGDClassifier` fit of the data, `GridSearchCV` was employed. `GridSearchCV` is a tool that takes a set of different hyperparameter values as input and finds the optimal combination out of these parameters that gives the best score according to a certain evaluation criterion, such as accuracy or F1 score, through an iterative process that tests every possible combination of

Table 4.2: This table lists the parameters that were selected for testing the SGDClassifier

Parameter	penalty	alpha	loss
Values	elasticnet	0.0001	log
	l2	0.0003	squared_hinge
	l1	0.0005	squared_loss
		0.001	huber
		0.005	
		0.01	

the given parameters to find the ideal combination. The CV in GridSearchCV stands for cross-validation, which refers to the process of splitting the dataset into several different training and testing datasets, and training the model several times, once for each training dataset split, then validating the results using the test dataset split. This cross-validation process ensures that the combination of parameters GridSearchCV chooses is likely to perform the best out of any other combination when used against new and unseen data, instead of only working situationally for a specific dataset. To find the optimal hyperparameters for the SGDClassifier, a GridSearchCV function for the penalty, alpha, and loss hyperparameters were used. The F1 and accuracy criteria were selected, and a 5 fold cross-validation was chosen. The resultant optimal combination of parameters was then retrieved using the `best_estimator_` and `best_score_` GridSearchCV methods. The hyperparameters and their values used in GridSearchCV are shown in Table 4.2.

### iii. SGD Classifier Best Fit Parameters and Performance Evaluation

The optimal parameters that produced the best F1 score and accuracy, which were found using GridSearchCV, are shown in Table 4.3, and the confusion matrix of the SGDClassifier test results can be seen in Figure 4.3, where the x-axis represents the predicted label and the y-axis represents the true label. The confusion matrix is a measure to visually evaluate the performance of a trained model. An ideal model with 100% accuracy produces a confusion matrix that has all its values on the diagonal, which represents a perfect match between the predicted and true labels. The farther away from the diagonal values are, the worse the

Table 4.3: The best score parameters for SGDClassifier that resulted from using GridSearchCV

Parameter	penalty	alpha	loss
Values	elasticnet	0.0005	squared_hinge

performance is. It can be seen here that, even with the best hyperparameters GridSearchCV could find, many predictions were not only wrong but missed the correct label by several levels. An example is how the model predicted 6 of the test undamaged dataset images as severely damaged (3). Furthermore, it also predicted 1 destroyed image as completely undamaged.

The best 5 fold cross-validation F1 score was 0.39 and the best accuracy achieved was 0.35. Due to the unsatisfactory SGDClassifier F1 score and accuracy results, other models were tested and explored.

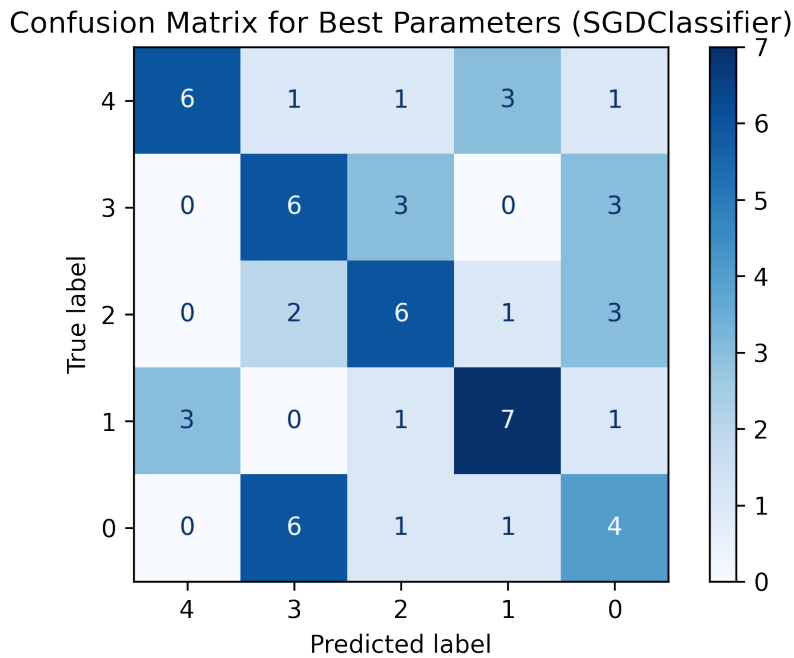


Figure 4.3: The confusion matrix for the best score parameters found from GridSearchCV using the parameters shown in Table 4.2. The labels shown in this figure refer to the following: 0 refers to undamaged, 1 refers to minor, 2 refers to moderate, 3 refers to severe, and 4 refers to destroyed

### 4.3.3 Support Vector Machines (SVC) Classifier

#### i. Overview and Method

The next classifier tested was the scikit-learn Support Vector Machines (SVM) classifier, also referred to as SVC. Based on the documentation of this classifier (scikit learn, 2020c), SVC is used for supervised learning classification. This classifier finds hyper-planes that separate the data into classes by transforming the data in high or infinite dimensions. The hyper-planes separate the data better when the distance, referred to as functional margin, to the nearest training data points is maximized. As the function margin increases, the generalization error of the classifier decreases. A penalty is imposed, using a hyperparameter called C, when a data point is classified incorrectly or if it falls within the functional margins. To find an optimal fit for the data, a balance between the functional margin and the C hyperparameter must be found, this allows the data points to be classified correctly but allows for some misclassification which ensures the data is not overfitted. For multi-class classification, SVC uses the one-versus-one strategy. To do this, the number of classifiers used is computed using the following equation:

$$\text{number of classifiers} = \text{number classes} \times \frac{(\text{number classes} - 1)}{2}$$

Each of these classifiers trains on data from two classes, hence the name one-versus-one. The name support vector refers to the data points that fall on the edge or within the functional margins, those points are called support vectors. Support vector machines in scikit-learn allows the use of several kernels. Kernels are tools that process the data in higher dimensions to find a hyper-plane that can separate the data into different classes. Kernels are user-specified when using SVC. For roof damage state classification, the Polynomial (poly) and the Radial Basis Function (RBF) kernels were tested. The basic idea of the poly kernel is that it transforms the data into higher dimensions using a user-specified hyperparameter called degree, which refers to the degree of the polynomial used to transform the data.

Transforming the data in this way allows a support vector classifier that separates the data to be found. The major hyperparameters that influence the performance of SVC when the poly kernel is selected are the degree, which refers to the degree of the polynomial, the coef0 hyperparameter, which refers to the coefficient of the selected polynomial, and the gamma hyperparameter, which controls the influence of each training data point. As gamma increases, the data points need to be closer for their classification to be affected. For the RBF kernel, it finds the hyper-plane that separates the data in infinite dimensions. This kernel could be described to behave similar to a weighted nearest neighbor model, which means that observations that are closer to a predicted observation have more influence on the classification than observations that are further. The RBF kernel has a hyperparameter gamma similar to the poly kernel, where it controls the influence of the distance observations have from each other on the classification of new observations. Additionally, SVC using either of these mentioned kernels are influenced by the C hyperparameter.

## **ii. Roof Damage Classification Using SVC Classifier**

To classify roof damage state using SVC, the same image processing steps described in the SGDClassifier section were used. Just as with SGDClassifier, the SVC inputs were two arrays, one array, X, consisting of all images in the training dataset in the form of floating-point arrays, and a second array, y, consisting of the labels associated with the images. Array X was standardized using the scikit-learn StandardScaler function here as well, which is recommended in the documentation of the SVC classifier. Next, the train\_test\_split function was used to split the data into 80% and 20% training and testing dataset, respectively. The shuffle and stratify options were again enabled. Next, to find the best hyperparameters for SVC, GridSearchCV was used here as well. The parameters that were tested are shown in Table 4.4.



Table 4.4: The best score parameters for SVC that resulted from using GridSearchCV.

Parameter	kernel	C	gamma	degree
Values	poly	0.1	auto	2
	rbf	1	scale	3
		10		5
		1000		10

Table 4.5: The best score parameters for SVC that resulted from using GridSearchCV

Parameter	kernel	C	gamma
Values	rbf	10	scale

### iii. SVC Classifier Best Fit Parameters and Performance Evaluation

The best score parameters resulting from GridSearchCV can be seen in Table 4.5. The best F1 score using 5 fold cross-validation was 0.54 and the best accuracy achieved was 0.55. The confusion matrix was plotted for the test dataset results and can be seen in Figure 4.4. From these performance evaluation measures, it can be seen that the SVC trained model performed much better than the SGD trained model. From the confusion matrix in Figure 4.4, it is observed that some predictions were not optimal. In particular, the trained SVC model tends to overestimate the predicted damage. This can be seen, for example, in the 6 roofs predicted to have severe roof damage state but the true label is undamaged roof damage state. These records make up 10% of the test data which is significant considering the difference between the true label and the predicted label is 3.

Overall, while an F1 score of 0.54 and accuracy of 0.55 for the trained SVC model are much greater than the SGDClassifier F1 score of 0.39 and accuracy of 0.35, these results still leave a large room for improvement. For this reason, one last classifier was tested to find better performance.

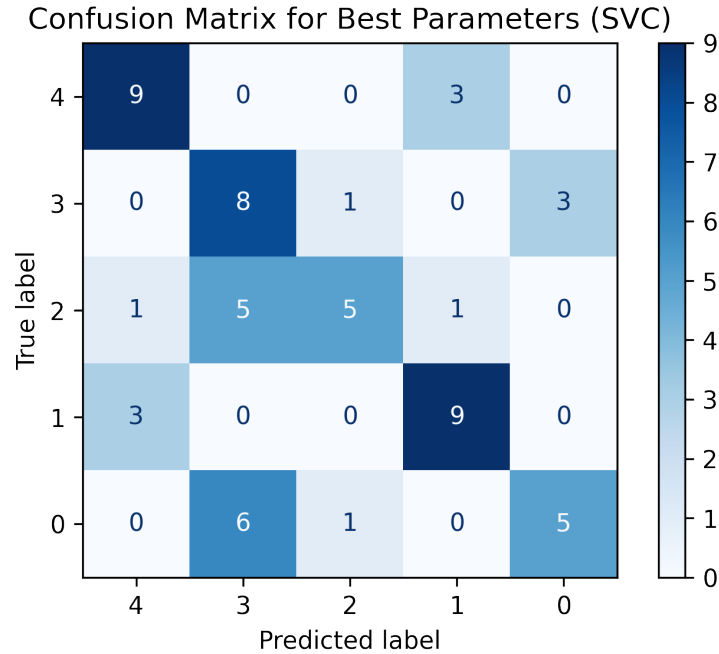


Figure 4.4: The confusion matrix for the best score parameters found from GridSearchCV using the parameters shown in Table 4.4. The labels shown in this figure refer to the following: 0 refers to undamaged, 1 refers to minor, 2 refers to moderate, 3 refers to severe, and 4 refers to destroyed

#### 4.3.4 Multi-layer Perceptron (MLP) Classifier

##### i. Overview and Method

After the SGDClassifier and the SVC models were tested and evaluated, the next and last model tested and evaluated was the scikit-learn Multi-layer Perceptron (MLP) MLP-Classifier. From the documentation of the classifier scikit learn (2020a), it is yet another supervised learning algorithm that is used for classification problems. The algorithm for this classifier can learn a nonlinear function approximator known as an activation function. Additionally, between the input and output layers, nonlinear layers are present, which are referred to as hidden layers. The input layer is comprised of several neurons, which pass the inputs through the activation function and sum the weights of the inputs to get a neuron output. It is called multi-layer perceptron because neurons are also referred to as perceptrons. For this classifier, there needs to be at least three layers, an input layer, an output layer, and at least

Table 4.6: The parameters tested for the MLPClassifier using GridSearchCV.

Parameter	activation	hidden_layer_sizes	solver	learning_rate
Values	relu	(100,)	adam	constant
	tanh	(50,100,)	sgd	adaptive
	logistic	(50,75,100,)	lbfgs	invscaling
	identity			

Table 4.7: The best score parameters for MLP that resulted from using GridSearchCV

Parameter	activation	hidden_layer_size	solver	learning_rate
Values	logistic	(100,)	adam	invscaling

one hidden layer. Hidden layers are mathematical functions that pass the weighted linear summation of the inputs into nonlinear functions that transform the data.. These layers are referred to as hidden because they are not visible to the system inputs and outputs. MLP uses backpropagation to train the classifier on the training data. Backpropagation refers to an algorithm that uses the gradient of a loss function taking into consideration the weight of each input and output data point. MLPClassifier has a regularization hyperparameter similar to the other models that were tested, this parameter helps avoid overfitting the training data. It also has many hyperparameters that help optimize the performance of the model.

## ii. Roof Damage Classification Using MLP Classifier

The process of implementing the MLPClassifier is similar to the previous two classifiers, where the classifier requires two arrays, an array X of images converted into floating-point values, and a second array y, that consists of the labels associated with the images in array X. Array X is scaled using the StandardScaler function, which is due to the MLPClassifier being sensitive to feature scaling similar to the SGDClassifier. The training dataset was split here as well into 80% training and 20% testing datasets. Next, GridSearchCV was used to find the optimal training parameters. The parameters that were tested can be seen in Table 4.6.

### iii. MLP Classifier Best Fit Parameters and Performance Evaluation

The best F1 score and accuracy parameters, which were found using GridSearchCV, can be seen in Table 4.7. The best 5 fold cross-validation F1 score being 0.49, slightly lower than the SVC score and the best accuracy was 0.47. Figure 4.5 shows the confusing matrix resulting from testing the MLPClassifier on the test dataset. Interestingly, while the trained SVC model tended to overestimate the damages, the trained MLPClassifier seems to slightly underestimate them in cases where the predictions were wrong. This is highlighted, for example, by the 7 images with a true label of severe, that were classified as undamaged by the MLP classifier.

One additional area to potentially enhance the performance was by first training a model to separate tarped roofs from untarped ones and see if this could then enhance the overall performance. This will be discussed next.

#### 4.3.5 Testing Tarped versus Untarped Roofs

Tarps are typically applied to damaged roofs after a windstorm event to protect the home from precipitation and other environmental elements. NOAA aerial imagery is typically captured a few days after the event. Due to this, some homes would have undergone repairs by then, and some roofs will be covered or partially covered by tarps. By examining the available data, it was noticed that the majority of the roofs in the training dataset with severe roof damage state are covered completely by tarps. When looking at the confusion matrices in Figure 4.3 and Figure 4.4, it can be seen that both trained models perform badly for the severe label. Both models misclassify other classes as severe as well as predicting other classes for that label. Due to this, one idea to try to solve this issue was to test if using binary classification to separate the roofs that are tarped from the roofs that are untarped and then training separate models for tarped and untarped roofs, would perform better and increase classification accuracy. The success of this idea depends completely on the ability to train a model that could reliably distinguish between tarped and untarped roofs with

exceptional accuracy since otherwise, misclassifications in this step would only add to the overall error and ultimately produce poorer performance scores. The highest precision score that was obtained from classifying tarped and untarped roofs was 82%, which shows there is potential to train a new model on tarped vs untarped roofs, but more research would be needed to associate tarp coverage with damage extent, which is beyond the scope of this study.

#### 4.3.6 Final Model Selection

Based on the results obtained from testing the three different classifiers combined with using GridSearchCV to find the best combination of the selected parameters, the SVM classifier SVC, with a 5 fold cross-validation score of 0.54 and accuracy of 0.55, was selected to use in the automation framework. A summary of the performance of all three tested models along with their accuracy and F1 score results can be seen in Table 4.8.

For the lower resolution NOAA imagery with a zoom level of 18, the same three classifiers were tested and GridSearchCV was used for this training dataset as well. The same steps were taken with the only difference being the number of images in the dataset and the number of pixels per image, which will change the size of each image array as well as the X array that consists of all images. The size of each image in the full resolution datasets was  $200 \times 200$  pixels, while the size of each image in the lower resolution dataset was  $50 \times 50$  pixels. The classifier that performed the best on this dataset is the MLPClassifier with the best 5 fold cross-validation F1 score being 0.31 and an accuracy of 0.30. The significantly worse performance (F1 percentage decrease of 42.6%) is attributed to the loss of essential details when smaller images are used. While these values are not satisfactory, due to time constraints, this trained model was still used for comparison with StEER's datasets for the Hurricane Michael (2018) dataset and the Nashville Tornadoes dataset, wherein both cases the resolution of NOAA aerial imagery was reduced to zoom level 18.

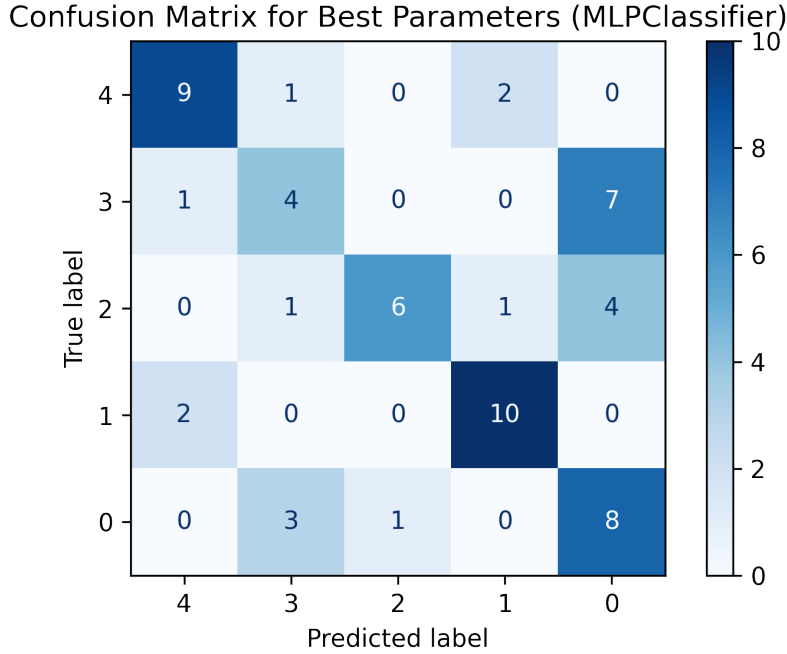


Figure 4.5: The confusion matrix for the best score parameters found from GridSearchCV using the parameters shown in Table 4.7. The labels shown in this figure refer to the following: 0 refers to undamaged, 1 refers to minor, 2 refers to moderate, 3 refers to severe, and 4 refers to destroyed

Table 4.8: Summary of the performance of the tested models to classify damage into five roof damage classes.

Model	SVC	SGD	MLP	MLP (Zoom 18)
<b>Accuracy</b>	0.55	0.35	0.47	0.30
<b>F1 Score</b>	0.54	0.39	0.49	0.31

#### 4.4 Framework Performance Evaluation and Comparison to Human Collected Data

To evaluate the performance of the automation framework, the framework was tested using 1562 records in total from StEER’s database. These records were taken from the following StEER datasets: 344 records from Hurricane Michael (2018) dataset, 792 records from the Nashville tornadoes dataset, and 426 records from Hurricane Laura (2020) dataset. The records were selected by filtering the data in Fulcrum using the criteria in Table 4.1 to find data that comply with these criteria. The data was also filtered to only include records

that had occupancy, number of stories, year built, roof type, and foundation type fields filled in, to ensure that the data resulting from the automation framework would be comparable. The comparisons are made relative to StEER’s data. However, it is important to note that StEER’s data, although expected to be highly reliable, cannot be taken as absolutely accurate. Table 4.9 shows the error percentages for each of the StEER datasets. These error percentages were calculated using a random sample of the dataset and reassessing the records to find the errors as described in Section 3.4. The addresses from this combined dataset were given as input for the automation framework. The following sections discuss the results of running the framework and the comparisons between the results and StEER’s datasets.

Table 4.9: Error percentages of StEER’s datasets used to evaluate the automation framework.

<b>Dataset</b>	<b>Hurricane Michael (2018)</b>	<b>Nashville Tornadoes</b>	<b>Hurricane Laura (2020)</b>
<b>Error (%)</b>	1.60%	2.01%	1.22%

#### 4.4.1 Web Scraping Vs. BRAILS AI Approach Vs. StEER Collected Data

The automation framework begins by taking the addresses from the combined StEER records dataset as inputs. Following the flowchart in Figure 4.1, the addresses are used in the web scraping process to find building attributes that consist of occupancy type, number of stories, year built, roof shape, and foundation type. Next, the coordinates are found using geocoding via Google API and are used to download Google satellite images as well as Google Street View images for each of the coordinates. These images are classified by BRAILS classifiers. The images that were downloaded in this step were manually inspected and the results were ignored for BRAILS roof type classifier if the downloaded satellite imagery did not show the roof clearly. The same was done for Street View imagery, where the results of BRAILS’ Occupancy classifier, Year Built classifier, and Number of Floors Detector were

ignored for any Street View imagery that was not clearly showing the structure, e.g., a tree was blocking the structure completely, or there was no imagery available.

Table 4.10 shows the number of results found per attribute as well as the percentage of the found data relative to the total number of records analyzed. From Table 4.10, it can be seen that some attributes are easier to find than others. For example, Occupancy and year built were among the highest successfully found attributes. This could be due to such data being readily available on realty websites, combined with the method used for web scraping. If other less-used search terms were used for the specific attributes, the results would have been different. On the other hand, roof shape and foundation type were particularly difficult to find and only a small percentage of the records had results for these attributes. This data is rarely listed on realty websites and could be found more commonly in county property tax websites which do not typically show up on the first page of Google results when searching for an address. If these websites were known and the data was not available in shapefile format from the county itself, the county websites could be used in web scraping instead of, or in addition to, Google search results. From the results summarized in Table 4.10, it can be seen that the web scraping approach used in the framework shows moderate success, especially when web scraping for readily available attributes, but leaves room for improvements, where more sophisticated web scraping code can be implemented to improve the accuracy for harder to find attributes.

Table 4.10: The number of results found per attribute using web scraping.

<b>Attribute</b>	<b>Occupancy</b>	<b>Number of Stories</b>	<b>Year Built</b>	<b>Roof Shape</b>	<b>Foundation Type</b>
<b>Number Found</b>	1456 (93%)	809 (52%)	1469 (94%)	231(15%)	133 (8%)

Next, the performance of all three methods in finding the building attributes is evaluated. The three methods are the human approach, represented by StEER’s data after the DE/QC process was completed, the web scraping method, and the AI approach represented



by BRAILS' classifiers results. For web scraping, the percentage relative to StEER's data represents the number of results that were found matching the entries in StEER's data. For BRAILS, the percentages relative to StEER's data represent the number of results that match the entries in StEER's data relative to the total number of records that consisted of reasonably clear images for the classifiers to predict from. To be able to conduct a fair comparison between the classes obtained from BRAILS classifiers and StEER's data, different labels for each field in the StEER dataset can be attributed to a single BRAILS prediction, as summarized in Table 4.12. It is important to note that these simplifications, while necessary for this comparison, represent a loss of granularity that may be useful for certain data reuse applications, and thus the simplifications are not ideal. An example of this would be the occupancy of the structure, where if there were only three options in the occupancy field: single-family residence, multi-family residence, and commercial or business, a manufactured home would likely fall into the single-family residence category. Manufactured homes are more structurally vulnerable than permanent single-family residence as noted by Strader et al. (2021), and this loss of valuable information would weaken analysis on data that does not include it.

Table 4.11 summarizes the percentage of attributes found by web scraping and through BRAILS that matched the attributes listed in the StEER dataset of the same records. The table highlights the strengths and weaknesses of each approach. For the occupancy attributes, web scraping excels with an accuracy percentage of 88.0%, while BRAILS performs poorly with a low accuracy percentage of 13.1%. For a better comparison, a confusion matrix showing the performance of BRAILS occupancy classifier relative to StEER's data can be seen in Figure 4.6, where SF refers to Single-Family, MF refers to Multi-Family, and B refers to Business/commercial. Furthermore, BRAILS results that were associated with a prediction probability of zero were removed when this confusion matrix was plotted. From this plot it can be seen that the majority of StEER's data that was tested was single-family,

where 64.8% were classified as business by BRAILS, 22% were classified as multi-family, and 12.9% as single-family.

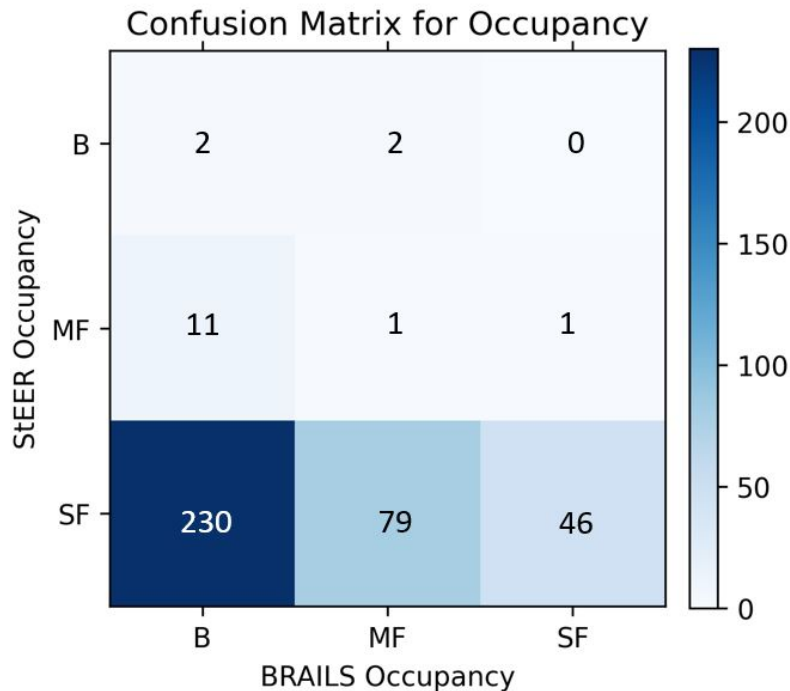


Figure 4.6: Confusion matrix showing the performance of BRAILS occupancy classifier relative to StEER’s data. SF refers to Single-Family, MF refers to Multi-Family, and B refers to Business/commercial.

The number of stories is another attribute that web scraping performed well in, with an accuracy percentage of 71.9%. BRAILS however has a slight edge here, with an even higher accuracy percentage of 73.6%. It is worth mentioning that for data that is primarily focused on residential structures, number of stories would typically range from 1-3, which would limit the number of results that can be found. For this reason, the comparative percentages are relatively high as there are fewer wrong answers to choose from. The year built attribute proves to be a challenge for both approaches, web scraping managed to reach a moderate accuracy percentage of 55.4%, while BRAILS accuracy percentage was 23.4%. Web scraping performed the worst for the roof shape attributes, where its accuracy percentage was only 22.1%. BRAILS on the other hand performed significantly better with a moderate accuracy percentage of 48.4%. Moreover, the roof shape field in StEER’s data consists of many

different options which can be seen in Table 4.12. Complex roof shapes, shown in Figure 4.7, are defined in StEER’s DE/QC handbook as roofs that consist of multiple roof shapes and typically have more than 10 ridgelines. When comparing BRAILS predictions relative to StEER’s data, the complex roofs in StEERs data were predicted as 78.9% hip roofs, 14% gable roofs, and 7.1% flat roofs. Additionally, BRAILS predictions for records in StEER’s data that had a gable roof were found to be 57.3% gable, 23.9% hip, and 18.8% flat. When checking BRAILS predictions for StEER’s records that had a hip roof, they were found to be 100% flat. For entries in StEER’s data of flat/hip combo option, the results were 14.2 % gable, 67.4% hip, and 18.4% flat.

Table 4.11: Comparison between human approach from StEER, web scraping, and BRAILS. The percentages are relative to StEER’s data.

Method	Attribute			
	Occupancy	Number of Stories	Year Built	Roof Shape
Web Scraping	88.0%	71.9%	55.4%	22.1%
BRAILS’ Classifiers	13.1%	73.6%	23.4%	48.4%



Figure 4.7: Examples of structures that have complex roofs based on StEER’s DE/QC handbook guidelines.

For visual comparison, an example of the obtained results from web scraping, BRAILS, and StEER’s data where the results did not match can be seen in Figure 4.8, where Figure 4.8a represents the image automatically captured from Google Street View and then classified

Table 4.12: StEER’s Occupancy field options and their equivalent BRAILS classes for the purpose of comparison.

<b>Occupancy</b>					
<b>StEER</b>	<b>BRAILS Equiva- lent</b>	<b>StEER</b>	<b>BRAILS Equiva- lent</b>	<b>StEER</b>	<b>BRAILS Equivalent</b>
Single Family	Single Family	Apartment	Multi Family	Assisted Living Center Business	Commercial
Manufactured Home Detached Garage Shed		Multi Family		Educational Factory and Industrial Government Hospital Office Professional Recreational Center Religious School Warehouse	

using BRAILS classifiers, Figure 4.8b represents an image of the same structure captured by the FAST during Hurricane Michael deployment. Web scraping could not yield any results for the number of stories of this single-family residence using the address, BRAILS classified this home as a 3 story residence using the Number of Stories Detector. On the other hand, StEER’s data listed this record as a two-story single-family home.

A second example can be seen in Figure 4.9, where Figure 4.9a represents the Google satellite imagery automatically downloaded and classified by BRAILS classifiers and Figure 4.9b represents an image of the same structure captured by the FAST during the Nashville Tornadoes deployment. For this record, web scraping found that the roof shape for this address is hip. On the other hand, BRAILS classified this roof shape as gable, which matches the entry in StEER’s data.

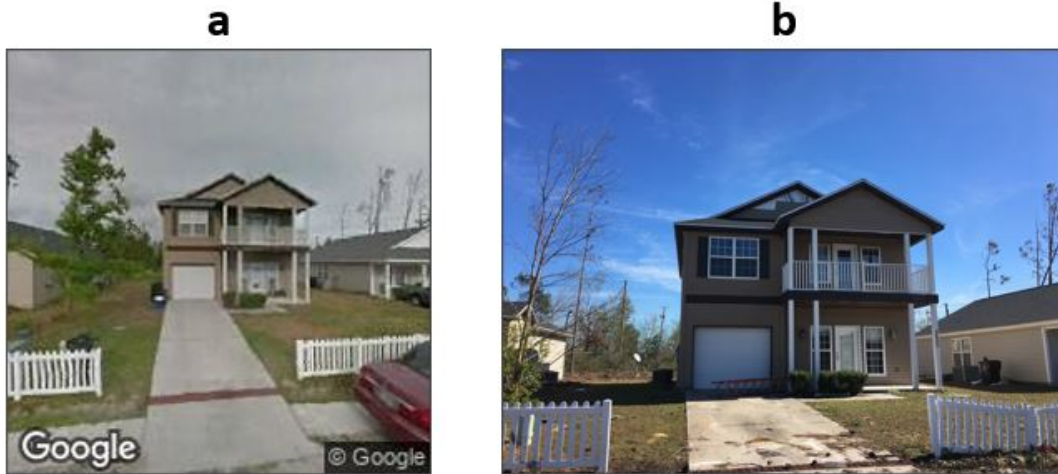


Figure 4.8: Two images of the same structure to compare results from web scraping, BRAILS, and StEER's data. a) An image downloaded automatically from Google Street View and used by BRAILS classifiers. b) An image of the same structure that was captured by the FAST deployed for Hurricane Michael.



Figure 4.9: Two images of the same structure to compare results from web scraping, BRAILS, and StEER's data. a) An image downloaded automatically from Google satellite imagery and classified by BRAILS classifiers. b) An image of the same structure that was captured by the FAST deployed for Nashville Tornadoes.

It is important to note in these comparisons that the BRAILS framework was developed for regional-scale hazard analysis as noted in (Wang et al., 2021) and is still under development and continuous improvements. The use of BRAILS classifiers in this work could be different than what it was intended for which could affect the results.

#### 4.4.2 Comparing SURF Enhanced Web Scraping Results with StEER's Data

As shown in Table 4.10, web scraping was not able to find all attributes for each of the addresses. For such cases, the gaps were filled using SURF, which learns the spatial patterns in the provided dataset and assigns attributes to the missing points based on those patterns. SURF trains a model based on input coordinates of the dataset and the labels associated with them, it then predicts an attribute for a given set of coordinates with unknown labels. SURF only accepts labels in numerical format, so the attribute values found from web scraping were temporarily converted to numbers to be used in SURF, then converted back to their original values once SURF filled the gaps. Table 4.13 shows the web scraping results before and after enhancing them with SURF relative to StEER's data. For the web scraped data before SURF enhancements, the percentages were calculated relative to the number of results that were found by web scraping. After the data was enhanced by SURF, all the gaps were filled in, so the number the correct results were divided by is larger than it was before. The SURF accuracies shown in Table 4.13 are all very close to the accuracy before the SURF enhancement, with a small overall decrease that does not exceed 3%. This shows that SURF is capable of filling gaps within a dataset no matter how large they are, while maintaining a very close overall accuracy that matches that of the original data supplied to the SURF model. Here, the web scraped foundation types were all the same value and thus SURF could not be used on them.

A final note on SURF utilization is that the data enhanced by SURF needed post-processing to remove some unreasonable values, such as the year built larger than 2021. For future incorporation of this open-source tool, some restraints should be applied to it to yield more accurate results.

#### 4.4.3 Comparing Damage Predictions to StEER's Data

As mentioned in Section 4.3.1, two final models were trained to classify roof damage state into five classes. One model was trained on the NOAA aerial imagery with zoom

Table 4.13: A comparison of web scraping results after enhancement with SURF with StEER’s data.

<b>Attribute</b>	<b>Occupancy</b>	<b>Number of Stories</b>	<b>Year Built</b>	<b>Roof Shape</b>
<b>Before SURF (%)</b>	88.0%	71.9%	55.4%	22.1%
<b>After SURF (%)</b>	87.3%	70.8%	52.4%	19.0%

level 20, and a second was trained on the NOAA aerial imagery with zoom level 18. The first model was used to predict the roof damage state of the records from Hurricane Laura (2020) dataset, while the second model was used to predict the roof damage state of the records from Hurricane Michael (2018) and Nashville Tornadoes datasets. Individual roof images were downloaded from NOAA aerial imagery for each of the wind events mentioned. Similar to the process of manual inspection of the downloaded images that were used for BRAILS classifiers, any image that was out of NOAA aerial coverage or if the imagery was not reasonably clear, the results were ignored and not included in the final analysis and comparison. 380 images were classified using the first model after manual inspection and 1063 were classified using the second trained model also after manual inspection. To be able to compare the results of the classification with StEER’s data, each of StEER’s records was assigned a roof damage state based on the criteria in Table 4.1. This was easily done because StEER uses component level damage ratios, which allows for straightforward implementation of any damage rating that is desired based on the component damage ratios. For each of the trained models, the predicted roof damage classification was compared to the roof damage classification assigned to StEER’s records based on their component level damage ratios. The first model correctly classified 197 images, which amounts to an accuracy of 51.8% and is comparable to the 5 fold cross-validated accuracy of the trained model (0.55). The second model correctly classified 326 images, which amounts to 30.6%. This is also comparable to the 5 fold cross-validated accuracy of the trained model (0.30).

Another measure of the performance of the trained model is the difference between the true label and the predicted label, which was found using the numerical values of each of the roof damage states, where undamaged is 0, minor is 1, moderate is 2, severe is 3, and destroyed is 4. Figure 4.10 represents a pie chart for the values of the difference between true label and predicted label, where for example, if the roof damage state was found to be undamaged (0) in StEER's data but the trained model predicted it as minor damage (1), the difference would be -1. The percentages present in the figure refer to the number of records that had a specific category of difference relative to all the records that were classified. From Figure 4.10, the majority of the misclassified images from the Hurricane Laura (2020) dataset consisted of a difference of 1 or -1. When inspecting the collected images that had a difference of 1 or -1, it can be seen that even for a human, it is difficult to see the damage in NOAA aerial imagery when the damage is minor. On the other hand, when inspecting the images of the same structure using the data sources available to StEER's Data Librarians, the damage can be easily detected. Figure 4.11 shows examples of images that were classified as undamaged roof damage state but their true label was minor damage. Figure 4.12 shows images of roofs that were classified as minor roof damage state but the true label was undamaged. Both figures show how difficult it can be to identify minor roof damage state from aerial imagery. Overall, the trained model performed very close to expectations based on the F1 score and accuracy that were found during the testing and training process. Additionally, if levels of error that are only one roof damage state above or below the ground truth label were allowed, the accuracy for this trained model would increase to 0.89.

Both F1 macro score and F1 micro scores were calculated manually for further evaluation. The F1 micro score is an overall F1 score and does not take into consideration label imbalance. On the other hand, the F1 macro score calculates the F1 score for each of the labels and calculates the mean of all the F1 scores found. The F1 micro score was found to be 0.52, which is similar to the F1 score found from 5 fold cross-validation during training



the model (0.54). The F1 macro score was found to be 0.25 and it can be seen that it is significantly less than the micro score. The reason for the lower macro score is due to the inability of the model to predict two roof damage states (severe and moderate). As can be seen evaluating the trained model in various ways is helpful in identifying weaknesses.

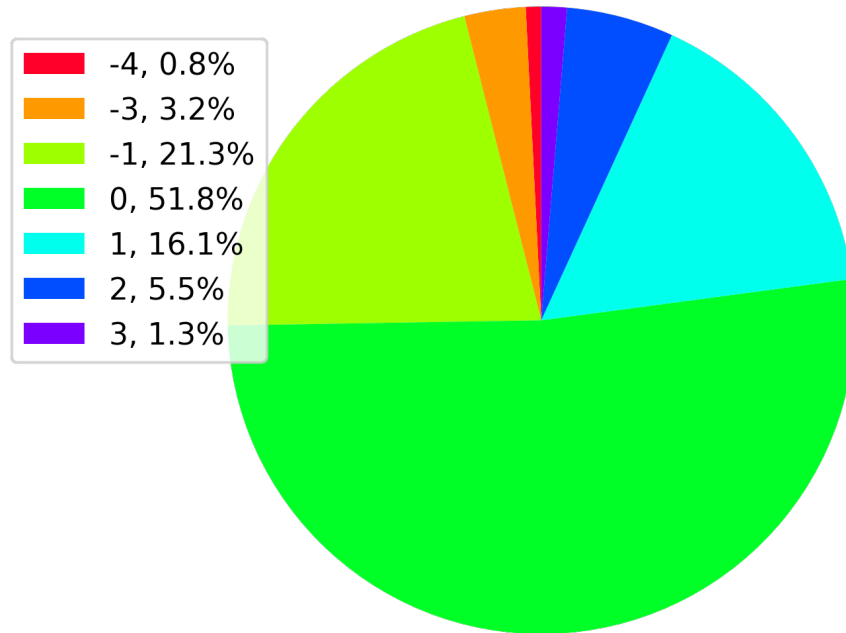


Figure 4.10: Pie chart that shows the difference between the true value and predicted value for the images classified using NOAA aerial imagery with zoom level 20 from Hurricane Laura (2020).



Figure 4.11: Roofs that were classified as undamaged but based on StEER's data are labeled as minor damage. These images show how easy it is to misclassify minor and undamaged roofs using aerial imagery only.

The results of the second model are represented in a similar way to the results of the first model, where the difference between the true label and the predicted label is illustrated in Figure 4.14. This pie chart shows the poor performance of this trained model. This



Figure 4.12: Roofs that were classified as minor but based on StEER’s data are labeled as undamaged. These images show how easy it is to misclassify minor and undamaged roofs using aerial imagery only.

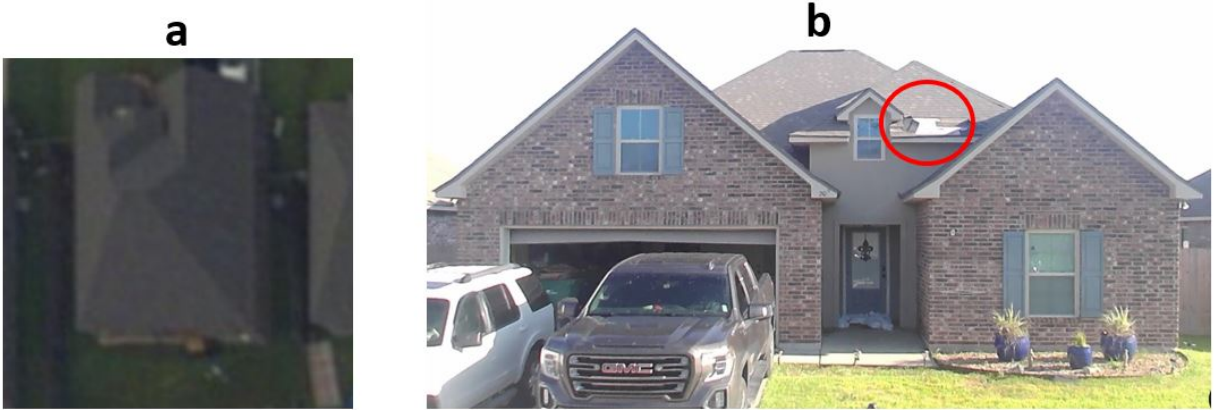


Figure 4.13: An example of a misclassified image. a) NOAA aerial imagery of the structure, which was predicted to be undamaged, b) An image of the same structure captured by the FAST, which was labeled as minor damage, where the red circle indicates the minor damage to the roof.

performance was expected based on the F1 score and the accuracy that were found when it was trained and tested. From Figure 4.14, the overall trend of the predictions is over predicting damage regardless of the class, as observed when comparing the percentages for all the negative values with the positive values in the figure, which indicates that the true label was less damaged than the predicted label. Both macro and micro F1 scores were found manually for this model as well. The micro F1 score was found to be 0.31, which is comparable to the 5 fold cross-validated F1 score in training. The macro F1 score was found to be 0.21, where the model was unable to predict any moderate roof damage state, which reduced the macro score. Another evaluation metric would be to allow levels of error that

are only one roof damage state above or below the ground truth label. If that level of error is allowed, the accuracy would increase to 0.51.

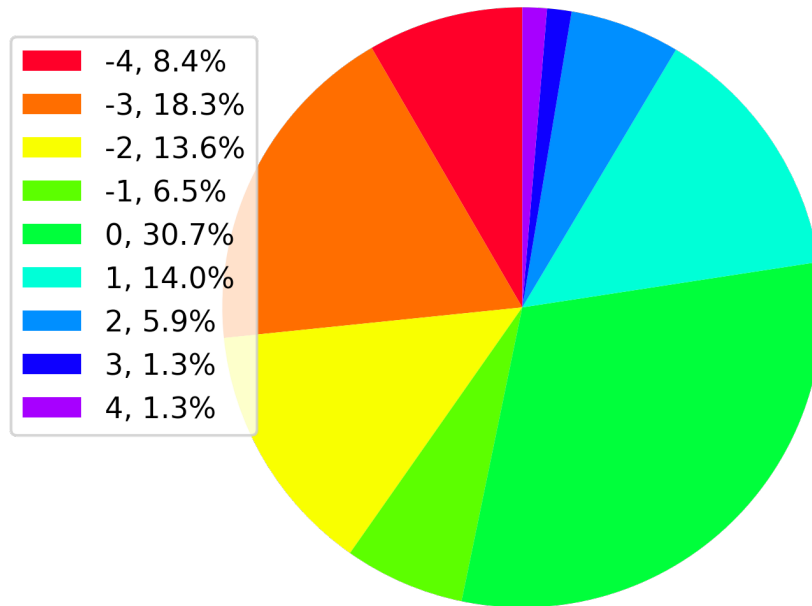


Figure 4.14: Pie chart that shows the difference between the true value and predicted value for the images classified using NOAA aerial imagery with zoom level 18 from Hurricane Michael (2018) and Nashville Tornadoes (2020).

#### 4.5 Future Work and Improvement

After looking at the results of running the framework on 1562 records and comparing the results with StEER’s data, areas that could use future improvement can be identified. While the web scraping results were acceptable for extracting occupancy and number of stories, this method was less reliable for extracting the remaining attributes. Other sources such as Zillow’s Assessors and Real Estate Database (ZTRAX), which is a reliable source to scrape building attributes, could be used instead of using a Google search.

The accuracy in roof damage state classification could be improved in the future by acquiring more data as well as having a better source for the images that are automatically captured. StEER collects imagery using UAS which are processed into orthomosaics and can then be uploaded to public repositories such as the OpenAerialMap platform. Such

aerial imagery could be used to automatically extract the images of individual roofs at higher resolutions in clusters where UAS was deployed. Additionally, the framework collects images pre-event as well as images post-event, which could be used to take a change detection approach instead of the supervised learning approach demonstrated here. Post-event street-level panoramic imagery could be automatically captured from street-level panoramic imagery collected by StEER's FAST and the pre-event imagery is already available in the current version of the framework. Moreover, an additional step could be added to eliminate manual checks for images captured from areas that did not have imagery coverage, which can be achieved either by using an additional machine learning step or checking the coordinates in an automated approach that does not involve machine learning. Adding this step would make the process more automated and efficient.

Implementing machine learning to classify roof damage status is just an example of how machine learning could be utilized in this preliminary framework. In future work, a supervised regression model could be trained to predict component level damage ratios to better match the direct component-level damage quantification preferred by StEER. To implement that, annotations for individual roofs and the damage they sustained might be required as well as using higher resolution images. Machine learning could also be used to detect other fields used in StEER's datasets, for example, to detect the presence of garage doors from the street-level panoramic imagery collected by StEER and uploaded to Mapillary. A script to automatically collect images of all visible sides using the parameters present in the links of the imagery in Mapillary could be added and a model that predicts binary classification can be trained on these images. Other attributes could be analyzed using these same images such as first-floor elevation, which could be done by training a supervised regression model. Additionally, the presence of fenestration protection could be predicted using street-level panoramic imagery by training a supervised binary classification model. Moreover, the foundation type field could be predicted using a supervised classification model trained on street-level panoramic from StEER or other sources.

The framework presented in this thesis, and the various modules incorporated within it, demonstrate how such technologies as web scraping and machine learning could automate parts of the data enhancement and quality control process. It provides a foundation that could be built upon in the future. The modular nature of the framework eases the process of updating various aspects of the module as well as adding additional functionality to it.

## 4.6 Chapter Summary

This chapter details the process of developing a preliminary framework that explores the incorporation of several modern technologies including web scraping and various types of machine learning. The framework takes a list of addresses or coordinates, then uses Web scraping to populate a building inventory dataset. Next, the two NHERI-SimCenter-developed python machine learning packages, BRAILS and SURF, were used to predict building attributes and fill in the gaps of the dataset generated by the framework. Finally, a scikit-learn SVC model trained to classify roof damage into five different damage states is used. The framework automatically downloads different types of imagery that are used in the various machine learning tasks, including Google Maps Satellite imagery, Google Street View imagery, as well as NOAA aerial imagery.

The chapter gives a background and discussion on each of the automation techniques and tools explored for the development of the framework. It also details the process of testing different scikit-learn machine learning models, including the SGD, SVC, and MLP classifiers, for roof damage classification. Finally, performance evaluation of the framework using 1562 records is detailed and discussed.

The topics covered and results discussed in the chapter highlight both the advantages and potential as well as the challenges and limitations that come with the utilization of such modern technologies in automating various parts the DE/QC process.

## Chapter 5

### Summary and Conclusion

This thesis introduces post-windstorm reconnaissance, highlighting its importance, describing its various processes, and introducing major contributors to it such as StEER. It was demonstrated that building performance datasets that result from post-windstorm field reconnaissance typically require extensive postprocessing to fill in the missing data and standardize existing data fields, and quality control to increase the accuracy of the data, to make them suitable for long-term reuse and knowledge discovery. The first contribution of this thesis describes a data enhancement and quality control process for post-windstorm reconnaissance data that, if applied consistently, will increase the quality and accuracy of building performance datasets over multiple events. StEER has adopted the DE/QC process described herein for windstorms, and is working to adapt the process to other hazard types. The DE/QC process is extensive and can take several months to complete. Thus, efforts to speed it up can be extremely valuable. The second contribution of this thesis is the introduction of a preliminary workflow that was built in Python to automate various components of the DE/QC process using modern technologies such as web scraping and machine learning.

The outcomes of this study are as follows:

- The data collected in post-windstorm reconnaissance is typically non-uniform, containing errors, gaps, and inconsistencies that if used in its raw state, would inhibit analysis and knowledge discovery. Thus, DE/QC protocols are essential.
- A formal DE/QC process was introduced and used on raw data from several recent windstorms. When this process was followed, high-quality, standardized, and accurate datasets were produced and published on DesignSafe-CI. This includes datasets for

Hurricane Michael with an error percentage of 1.60%, the Nashville Tornadoes with an error percentage of 2.01%, and Hurricane Laura with an error percentage of 1.22%.

- A framework for training undergraduate students on the DE/QC process was developed. Several students were successfully trained as Data Librarians who worked on the aforementioned windstorms.
- A preliminary framework was developed to demonstrate how various aspects of the DE/QC process can be automated using web scraping and machine learning. The framework was tested on 1562 addresses from StEER's windstorm database and its performance on these records was evaluated. The framework was specifically used to (1) populate building attributes using web scraping and the two deep learning packages BRAILS and SURF, and (2) classify the roof damage state for each address.
- After testing the automation framework and comparing its results relative to the records in the StEER database, it was found that web scraping achieved the following accuracies: 88.0% for occupancy, 71.9% for number of stories, 55.4% for year built, and 22.1% for roof shape. BRAILS classifiers achieved the following accuracies: 13.1% for occupancy, 73.6% for number of stories, 23.4% for year built, and 48.4% for the roof shape. SURF was shown to enhance a dataset by filling in the gaps while maintaining a close accuracy to that of the original data fed to the SURF model for enhancement.
- The effects of aerial imagery resolution in automated recognition and classification of roof damage was quantified. The SVC model that was trained to predict the roof damage state for the higher-resolution images had an accuracy of 0.52, while the MLP model trained on the lower-resolution images had a significantly lower accuracy of 0.31. This was attributed to the loss of important features as the image quality was reduced.

The final qualities of the building performance datasets highlight the importance of developing and implementing a comprehensive DE/QC process. The manually processed

datasets provide robust, high-quality, benchmarks for validating machine-augmented approaches, and ultimately provide datasets that are ideally suited for extracting knowledge that will advance hazards engineering and strengthen the resilience of at-risk communities. The results from the preliminary automation framework demonstrate the viability of exploring modern technologies to speed up the DE/QC process, and show promise that with further improvements to the web scraping and machine learning algorithms, such techniques could augment the human approach. However, the common methods employed in this thesis showed mixed results in terms of being able to match the accuracy of carefully human-labeled approaches.



## Bibliography

- Smith B. Adam. U.S. Billion-dollar Weather and Climate Disasters, 1980 - present (NCEI Accession 0209268), 2020. URL <https://doi.org/10.25921/stkw-7w73>.
- Beverley J Adams, Charles K Huyck, Babak Mansouri, Ronald T Eguchi, and Masanobu Shinozuka. Application of high-resolution optical satellite imagery for post-earthquake damage assessment: The 2003 boumerdes (algeria) and bam (iran) earthquakes. *Research Progress and Accomplishments 2003-2004, Buffalo: MCEER*, 2004.
- Beverley J. Adams, Babak Mansouri, and Charles K. Huyck. Streamlining post-earthquake data collection and damage assessment for the 2003 bam, iran, earthquake using views™ (visualizing impacts of earthquakes with satellites). *Earthquake Spectra*, 21(1-suppl):213–218, 2005. doi: 10.1193/1.2098588. URL <https://doi.org/10.1193/1.2098588>.
- Mohammad Aghababaei, Maria Koliou, and Stephanie G. Paal. Performance assessment of building infrastructure impacted by the 2017 hurricane harvey in the port aransas region. *Journal of Performance of Constructed Facilities*, 32(5):04018069, 2018. doi: 10.1061/(ASCE)CF.1943-5509.0001215. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CF.1943-5509.0001215>.
- Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine learning from theory to algorithms: An overview. *Journal of Physics: Conference Series*, 1142:012012, nov 2018. doi: 10.1088/1742-6596/1142/1/012012. URL <https://doi.org/10.1088/1742-6596/1142/1/012012>.

Mehrshad Amini and Ali M. Memari. Review of literature on performance of coastal residential buildings under hurricane conditions and lessons learned. *Journal of Performance of Constructed Facilities*, 34(6):04020102, 2020. doi: 10.1061/(ASCE)CF.1943-5509.0001509. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CF.1943-5509.0001509>.

ATC. Detailed evaluation safety assessment form, 2005a.

ATC. Rapid evaluation safety assessment form, 2005b.

Julius Baier. Wind pressure in the st. louis tornado, with special reference to the necessity of wind bracing for high buildings. *Transactions of the American Society of Civil Engineers*, 37(1):221–289, 1897. doi: 10.1061/TACEAT.0001268. URL <https://ascelibrary.org/doi/abs/10.1061/TACEAT.0001268>.

Jeffrey W. Berman, Joseph Wartman, Michael Olsen, Jennifer L. Irish, Scott B. Miles, Troy Tanner, Kurtis Gurley, Laura Lowes, Ann Bostrom, Jacob Dafni, Michael Grilliot, Andrew Lyda, and Jaqueline Peltier. Natural hazards reconnaissance with the nheri rapid facility. *Frontiers in Built Environment*, 6:185, 2020. ISSN 2297-3362. doi: 10.3389/fbuil.2020.573067. URL <https://www.frontiersin.org/article/10.3389/fbuil.2020.573067>.

James Bialas, Thomas Oommen, Umaa Rebbapragada, and Eugene Levin. Object-based classification of earthquake damage from high-resolution optical imagery using machine learning. *Journal of Applied Remote Sensing*, 10(3):1 – 16, 2016. doi: 10.1117/1.JRS.10.036025. URL <https://doi.org/10.1117/1.JRS.10.036025>.

Cheryl Ann Blain, Antonio Bobet, JoAnn Browning, Billy L. Edge, William Holmes, David R. Johnson, Marti LaChance, Julio Ramirez, Ian Robertson, Tom Smith, Chris Thompson, Karina Vielma, Dan Zehner, and Delong Zuo. The network coordination office of nheri (natural hazards engineering research infrastructure). *Frontiers in*

*Built Environment*, 6:108, 2020. ISSN 2297-3362. doi: 10.3389/fbuil.2020.00108. URL <https://www.frontiersin.org/article/10.3389/fbuil.2020.00108>.

Jay H. Crandell and Vladimir Kochkin. Scientific damage assessment methodology and practical applications. In *Structures Congress 2005*, pages 1–12, 2005. doi: 10.1061/40753(171)248. URL <https://ascelibrary.org/doi/abs/10.1061/40753%28171%29248>.

P. Shane Crawford, Alexander M. Hainen, Andrew J. Graettinger, John W. van de Lindt, and Lawrence Powell. Discrete-outcome analysis of tornado damage following the 2011 tuscaloosa, alabama, tornado. *Natural Hazards Review*, 21(4):04020040, 2020. doi: 10.1061/(ASCE)NH.1527-6996.0000396. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29NH.1527-6996.0000396>.

Alyssa C. Egnew, David B. Roueche, and David O. Prevatt. Linking building attributes and tornado vulnerability using a logistic regression model. *Natural Hazards Review*, 19(4):04018017, 2018. doi: 10.1061/(ASCE)NH.1527-6996.0000305. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29NH.1527-6996.0000305>.

FEMA. Fema 356 prestandard and commentary for the seismic rehabilitation of building. *Rehabilitation*, 221, 2000.

FEMA. Standard operating procedures for mitigation assessment team process, 2008.

FEMA. Building Science Support and Code Changes Aiding Sandy Recovery, 2014.

FEMA. Hurricane Irma in Florida, 2018.

FEMA. Hurricane Harvey in Texas, 2019.

FEMA. Hurricane Michael in Florida, 2020a.

FEMA. Openfema, 2020b. URL <https://www.fema.gov/about/reports-and-data/openfema>.

- Christopher Fox, Anany Levitin, and Thomas Redman. The notion of data and its quality dimensions. *Information Processing & Management*, 30(1):9 – 19, 1994. ISSN 0306-4573. doi: [https://doi.org/10.1016/0306-4573\(94\)90020-5](https://doi.org/10.1016/0306-4573(94)90020-5). URL <http://www.sciencedirect.com/science/article/pii/0306457394900205>.
- J. Carol Friedland. *Residential building damage from hurricane storm surge: proposed methodologies to describe, assess and model building damage*. Dissertation, Louisiana State University, 2009.
- P. Fronstin, A. Holtmann, and Coral Gables. The determinants of residential property damage caused by hurricane andrew. *Southern Economic Journal*, 61:387, 1994.
- T. Theodore Fujita. *Plainfield Tornado of August 28, 1990*, pages 1–17. American Geophysical Union (AGU), 1993. ISBN 9781118664148. doi: <https://doi.org/10.1029/GM079p0001>. URL <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/GM079p0001>.
- Shubharoop Ghosh, Beverley Adams, J. Womble, Carol Friedland, and Ronald Eguchi. Deployment of remote sensing technology for multi-hazard post-katrina damage assessment. In *Fourth International Workshop on Remote Sensing for Post Disaster Response Cambridge, UK*, 01 2021.
- John Gross, Joseph Main, Long Phan, Fahim Sadek, Stephen Cauffman, and David Jorgensen. Final report on the collapse of the dallas cowboys indoor practice facility, may 2, 2009 (nist ir 7661), 2010-01-26 2010. URL [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=904696](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=904696).
- K. R. Gurley and F. J. Masters. Post-2004 hurricane field survey of residential building performance. *Natural Hazards Review*, 12(4):177–183, 2011. doi: 10.1061/(ASCE)NH.1527-6996.0000044. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29NH.1527-6996.0000044>.

Shahid Hamid, B.M. Golam Kibria, Sneha Gulati, Mark Powell, Bachir Annane, Steve Cocke, Jean-Paul Pinelli, Kurt Gurley, and Shu-Ching Chen. Predicting losses of residential structures in the state of florida by the public hurricane loss evaluation model. *Statistical Methodology*, 7(5):552 – 573, 2010. ISSN 1572-3127. doi: <https://doi.org/10.1016/j.stamet.2010.02.004>. URL <http://www.sciencedirect.com/science/article/pii/S1572312710000195>.

Adam Hatzikyriakou and Ning Lin. Assessing the vulnerability of structures and residential communities to storm surge: An analysis of flood impact during hurricane sandy. *Frontiers in Built Environment*, 4:4, 2018. ISSN 2297-3362. doi: 10.3389/fbuil.2018.00004. URL <https://www.frontiersin.org/article/10.3389/fbuil.2018.00004>.

Craig Henderson, Tim Huff, and Gary Bouton. Structural observations and tornado damage mitigation concepts: March 2020 tennessee tornadoes. *Practice Periodical on Structural Design and Construction*, 26(2):05021001, 2021. doi: 10.1061/(ASCE)SC.1943-5576.0000571. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29SC.1943-5576.0000571>.

Anant Jain, Arpit A Bhusar, David B Roueche, and David O Prevatt. Engineering-based tornado damage assessment: Numerical tool for assessing tornado vulnerability of residential structures. *Frontiers in Built Environment*, 6:89, 2020.

Mohammad Kakooei and Yasser Baleghi. Fusion of satellite, aircraft, and uav data for automatic disaster damage assessment. *International Journal of Remote Sensing*, 38(8-10):2511–2534, 2017. doi: 10.1080/01431161.2017.1294780. URL <https://doi.org/10.1080/01431161.2017.1294780>.

Jian Kang, Marco Körner, Yuanyuan Wang, Hannes Taubenböck, and Xiao Xiang Zhu. Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:44–59, 2018. ISSN 0924-2716. doi: <https://doi.org/10.1016/>

j.isprsjprs.2018.02.006. URL <https://www.sciencedirect.com/science/article/pii/S0924271618300352>. Deep Learning RS Data.

Ahsan Kareem. Structural performance and wind speed damage correlation in hurricane alicia. *Journal of Structural Engineering*, 111(12):2596–2610, 1985. doi: 10.1061/(ASCE)0733-9445(1985)111:12(2596). URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9445%281985%29111%3A12%282596%29>.

Alireza G. Kashani, Patrick S. Crawford, Sufal K. Biswas, Andrew J. Graettinger, and David Grau. Automated tornado damage assessment and wind speed estimation based on terrestrial laser scanning. *Journal of Computing in Civil Engineering*, 29(3):04014051, 2015. doi: 10.1061/(ASCE)CP.1943-5487.0000389. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CP.1943-5487.0000389>.

Edward L. Keith and John D. Rose. Hurricane andrew structural performance of buildings in south florida. *Journal of Performance of Constructed Facilities*, 8(3):178–191, 1994. doi: 10.1061/(ASCE)0887-3828(1994)8:3(178). URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290887-3828%281994%298%3A3%28178%29>.

Andrew Kennedy, Andrew Copp, Matthew Florence, Anderson Gradel, Kurtis Gurley, Matt Janssen, James Kaihatu, Douglas Krafft, Patrick Lynett, Margaret Owensby, Jean-Paul Pinelli, David O. Prevatt, Spencer Rogers, David Roueche, and Zachariah Silver. Hurricane michael in the area of mexico beach, florida. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 146(5):05020004, 2020. doi: 10.1061/(ASCE)WW.1943-5460.0000590. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29WW.1943-5460.0000590>.

Tracy Kijewski-Correa, Nathan Smith, Alexandros Taflanidis, Andrew Kennedy, Cheng Liu, Markus Krusche, and Charles Vardeman. Cybereye: Development of integrated cyber-infrastructure to support rapid hurricane risk assessment. *Journal of Wind Engineering*

*and Industrial Aerodynamics*, 133:211 – 224, 2014. ISSN 0167-6105. doi: <https://doi.org/10.1016/j.jweia.2014.06.003>. URL <http://www.sciencedirect.com/science/article/pii/S016761051400110X>.

Tracy Kijewski-Correa, David Prevatt, Ian Robertson, Daniel Smith, Khalid Mosalam, David Roueche, Benjamin Lichty, Mariantonieta Gutierrez Soto, Aly Mousaad Aly, Ali Lenjani, et al. Steer-hurricane michael: Preliminary virtual assessment team (p-vat) report, 2018.

Tracy Kijewski-Correa, David Roueche, Khalid M. Mosalam, David O. Prevatt, and Ian Robertson. Steer: A community-centered approach to assessing the performance of the built environment after natural hazard events.manuscript submitted for publication.). *Frontiers in Built Environment*, 2021.

G. Kong and H. Fan. Enhanced facade parsing for street-level images using convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–13, 2020. doi: 10.1109/TGRS.2020.3035878.

Erica Kuligowski, Long Phan, Marc Levitan, and David Jorgensen. Preliminary reconnaissance of the may 20, 2013, newcastle-moore tornado in oklahoma, 2013-12-03 2013.

Erica Kuligowski, Franklin Lombardo, Long Phan, Marc Levitan, and David Jorgensen. Final report, national institute of standards and technology (nist) technical investigation of the may 22, 2011, tornado in joplin, missouri, 2014-03-26 2014.

VS Lakshminarasimhan. *Image based assessment of windstorm damage*. PhD thesis, Thesis, Texas Tech University, Texas, 2004.

Andrew Cowper Lawson. *The California earthquake of April 18, 1906: Report of the state earthquake investigation commission*. Carnegie institution of Washington, 1908.

- Hai Lew, Emil Simiu, and John Gross. Manual for seismic and windstorm evaluation of existing concrete buildings for dominican republic (nist ir 6867), 2002-04-01 2002. URL [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=860413](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=860413).
- Yan Li, Yiqun Chen, Abbas Rajabifard, and Kourosh Khoshelham. Estimating building age from google street view images using deep learning, 09 2018a.
- Yue Li and Bruce R. Ellingwood. Hurricane damage to residential construction in the us: Importance of uncertainty modeling in risk assessment. *Engineering Structures*, 28(7): 1009 – 1018, 2006. ISSN 0141-0296. doi: <https://doi.org/10.1016/j.engstruct.2005.11.005>. URL <http://www.sciencedirect.com/science/article/pii/S0141029605004335>.
- Yundong Li, Shi Ye, and Ivan Bartoli. Semisupervised classification of hurricane damage from postevent aerial imagery using deep learning. *Journal of Applied Remote Sensing*, 12(4):1 – 13, 2018b. doi: 10.1117/1.JRS.12.045008. URL <https://doi.org/10.1117/1.JRS.12.045008>.
- Joseph Main, Maria Dillard, Erica Kuligowski, Benjamin Davis, Jazalyn Dukes, Kenneth Harrison, Jennifer Helgeson, Katherine Johnson, Marc Levitan, Judith Mitrani-Reiser, Scott Weaver, DongHun Yeo, Luis, Joel Cline, and Thomas Kirsch. Learning from hurricane maria’s impacts on puerto rico: A progress report, 2021-01-19 2021.
- Timothy P. Marshall. Tornado damage survey at moore, oklahoma. *Weather and Forecasting*, 17(3):582 – 598, 2002. doi: 10.1175/1520-0434(2002)017<0582:TDSAMO>2.0.CO;2. URL [https://journals.ametsoc.org/view/journals/wefo/17/3/1520-0434\\_2002\\_017\\_0582\\_tdsamo\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/wefo/17/3/1520-0434_2002_017_0582_tdsamo_2_0_co_2.xml).
- Timothy P Marshall, JR McDonald, and GS Forbes. The enhanced fujita (ef) scale. In *Preprints, 22nd Conf. on Severe Local Storms, Hyannis, MA, Amer. Meteor. Soc. B*, volume 3, 2004.



Hassan Masoomi, Mohammad R. Ameri, and John W. van de Lindt. Wind performance enhancement strategies for residential wood-frame buildings. *Journal of Performance of Constructed Facilities*, 32(3):04018024, 2018. doi: 10.1061/(ASCE)CF.1943-5509.0001172. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CF.1943-5509.0001172>.

Carol C. Massarra, Friedland Carol J., Marx Brian D., and Dietrich J. Casey. Multihazard hurricane fragility model for wood structure homes considering hazard parameters and building attributes interaction. *Frontiers in Built Environment*, 6:147, 2020. ISSN 2297-3362. doi: 10.3389/fbuil.2020.00147.

A McMillan, BJ Adams, A Reynolds, T Brown, D Liang, and A Womble. Mceer response: Advanced technology for rapid tornado damage assessment following the ‘super tuesday’ tornado outbreak of february 2008. Technical report, MCEER-08-SP01, MCEER, University at Buffalo. Available online at: [http . . .](http://www.mceer.buffalo.edu/research/rapid-response/), 2008.

Microsoft. Computer generated building footprints for the united states, 2018. URL <https://github.com/microsoft/USBuildingFootprints>.

J. E. Minor, J. R. McDonald, and K. C. Mehta. The tornado: An engineering-oriented perspective. NASA STI/Recon Technical Report N, December 1977.

Spatial Networks. Fulcrum App for Android and IOS (Release Version 2.41)., 2021.

K. R. Nia and G. Mori. Building damage assessment using deep learning and ground-level image data. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 95–102, 2017. doi: 10.1109/CRV.2017.54.

NOAA. Record-breaking atlantic hurricane season draws to an end, 2020. URL <https://www.noaa.gov/media-release/record-breaking-atlantic-hurricane-season-draws-to-end>.

- Hyongsu Park, Tori Tomiczek, Daniel T. Cox, John W. van de Lindt, and Pedro Lomonaco. Experimental modeling of horizontal and vertical wave forces on an elevated coastal structure. *Coastal Engineering*, 128:58 – 74, 2017. ISSN 0378-3839. doi: <https://doi.org/10.1016/j.coastaleng.2017.08.001>. URL <http://www.sciencedirect.com/science/article/pii/S0378383917300650>.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Lori Peek, Jennifer Tobin, Rachel M. Adams, Haorui Wu, and Mason Clay Mathews. A framework for convergence research in the hazards and disaster field: The natural hazards engineering research infrastructure converge facility. *Frontiers in Built Environment*, 6:110, 2020. ISSN 2297-3362. doi: [10.3389/fbuil.2020.00110](https://doi.org/10.3389/fbuil.2020.00110). URL <https://www.frontiersin.org/article/10.3389/fbuil.2020.00110>.
- Jean-Paul Pinelli, E. Simiu, Kurtis Gurley, Chelakara Subramanian, Liang Zhang, Anne Cope, James Filliben, and Shahid Hamid. Hurricane damage prediction model for residential structures. *Journal of Structural Engineering-asce - J STRUCT ENG-ASCE*, 130, 11 2004. doi: [10.1061/\(ASCE\)0733-9445\(2004\)130:11\(1685\)](https://doi.org/10.1061/(ASCE)0733-9445(2004)130:11(1685)).
- Aimilia K Pistrika and Sebastiaan N Jonkman. Damage to residential buildings due to flooding of new orleans after hurricane katrina. *Natural Hazards*, 54(2):413–434, 2010.
- David O. Prevatt, William Coulbourne, Andrew J. Graettinger, Shiling Pei, Rakesh Gupta, and David Grau. *Joplin, Missouri, Tornado of may 22, 2011: Structural damage survey and case for tornado-resilient building codes*. American Society of Civil Engineers (ASCE), January 2012a. ISBN 9780784412503. doi: [10.1061/9780784412503](https://doi.org/10.1061/9780784412503).

David O. Prevatt, William Coulbourne, Andrew J. Graettinger, Shiling Pei, Rakesh Gupta, and David Grau. *Joplin, Missouri, Tornado of May 22, 2011*. American Society of Civil Engineers, 2012b. doi: 10.1061/9780784412503. URL <https://ascelibrary.org/doi/abs/10.1061/9780784412503>.

David O. Prevatt, John W. van de Lindt, Edward W. Back, Andrew J. Graettinger, Shiling Pei, William Coulbourne, Rakesh Gupta, Darryl James, and Duzgun Agdas. Making the case for improved structural design: Tornado outbreaks of 2011. *Leadership and Management in Engineering*, 12(4):254–270, 2012c. doi: 10.1061/(ASCE)LM.1943-5630.0000192. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29LM.1943-5630.0000192>.

Sudha Radhika, Yukio Tamura, and Masahiro Matsui. Cyclone damage detection on building structures from pre- and post-satellite images using wavelet based pattern recognition. *Journal of Wind Engineering and Industrial Aerodynamics*, 136:23 – 33, 2015. ISSN 0167-6105. doi: <https://doi.org/10.1016/j.jweia.2014.10.018>. URL <http://www.sciencedirect.com/science/article/pii/S0167610514002207>.

Chris Ramseyer, Lisa Holliday, and Royce Floyd. Enhanced residential building code for tornado safety. *Journal of Performance of Constructed Facilities*, 30(4):04015084, 2016. doi: 10.1061/(ASCE)CF.1943-5509.0000832. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CF.1943-5509.0000832>.

Ellen M. Rathje, Clint Dawson, Jamie E. Padgett, Jean-Paul Pinelli, Dan Stanzione, Pedro Arduino, Scott J. Brandenberg, Tim Cockerill, Maria Esteva, Fred L. Haan, Ahsan Kareem, Laura Lowes, and Gilberto Mosqueda. Enhancing research in natural hazards engineering through the designsafe cyberinfrastructure. *Frontiers in Built Environment*, 6:213, 2020. ISSN 2297-3362. doi: 10.3389/fbuil.2020.547706. URL <https://www.frontiersin.org/article/10.3389/fbuil.2020.547706>.

- Harry Fielding Reid. The mechanics of the earthquake, the california earthquake of april 18, 1906; report of the state investigation commission, 1910.
- Michael Riley. Reconnaissance report on damage to engineered structures during the may 1999 oklahoma city tornado, 2002-08-01 2002.
- D. Roueche, S. Kameshwar, J. Marshall, N. Mashrur, T. Kijewski-Correa, K. Gurley, I. Afanasyeva, G. Brasic, J. Cleary, D. Golovichev, O. Lafontaine, F. Lombardo, L. Micheli, B. Phillips, D. Prevatt, I. Robertson, J. Schroeder, D. Smith, S. Strader, M. Wilson, K. Ambrose, H. Rawajfih, and L Rodriguez. Hybrid preliminary virtual reconnaissance report-early access reconnaissance report (pvrr-earr), 2021.
- David Roueche. Virtual assessment structural team (vast) handbook: Data enrichment and quality control (de/qc) for us windstorms, 2019. URL <https://drive.google.com/file/d/1-ZS7oJbfHooj9m00cz-ewbqjDCfeUxDL/view>.
- David B. Roueche, Franklin T. Lombardo, and David O. Prevatt. Empirical approach to evaluating the tornado fragility of residential structures. *Journal of Structural Engineering*, 143(9):04017123, 2017. doi: 10.1061/(ASCE)ST.1943-541X.0001854. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29ST.1943-541X.0001854>.
- David B. Roueche, Franklin T. Lombardo, Daniel J. Smith, and Richard J. Krupar. Fragility assessment of wind-induced residential building damage caused by hurricane harvey, 2017. *Forensic Engineering 2018*, pages 1039–1048, 2018. doi: 10.1061/9780784482018.100. URL <https://ascelibrary.org/doi/abs/10.1061/9780784482018.100>.
- R. Saravanan and P. Sujatha. A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949, 2018. doi: 10.1109/ICCONS.2018.8663155.

scikit learn. sklearn.neural\_network.mlpclassifier, 2020a. URL [https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html).

scikit learn. sklearn.linear\_model.sgdclassifier, 2020b. URL [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.SGDClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html).

scikit learn. sklearn.svm.svc, 2020c. URL <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.

selenium, 2018.

Dakshina Silva, Jamie Kruse, and Yongsheng Wang. Spatial dependencies in wind-related housing damage. *Natural Hazards: Journal of the International Society for the Prevention and Mitigation of Natural Hazards*, 47(3):317–330, December 2008. URL <https://ideas.repec.org/a/spr/nathaz/v47y2008i3p317-330.html>.

B. Sirmacek and C. Unsalan. Damaged building detection in aerial images using shadow information. In *2009 4th International Conference on Recent Advances in Space Technologies*, pages 249–252, 2009. doi: 10.1109/RAST.2009.5158206.

Stephen M. Strader, David B. Roueche, and Brett M. Davis. Unpacking tornado disasters: Illustrating southeastern us tornado mobile and manufactured housing problem using march 3, 2019 beauregard-smith station, alabama, tornado event. *Natural Hazards Review*, 22(1):04020060, 2021. doi: 10.1061/(ASCE)NH.1527-6996.0000436. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29NH.1527-6996.0000436>.

Mason Taggart and John W. van de Lindt. Performance-based design of residential wood-frame buildings for flood based on manageable loss. *Journal of Performance of Constructed Facilities*, 23(2):56–64, 2009. doi: 10.1061/(ASCE)0887-3828(2009)23:2(56). URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290887-3828%282009%2923%3A2%2856%29>.

Harvey Thurm Taylor, Bill Ward, Mark Willis, and Walt Zaleski. The saffir-simpson hurricane wind scale. *Atmospheric Administration: Washington, DC, USA*, 2010.

EagleView Technologies. Pictometry aerial imagery , 2021.

J. Thomas, A. Kareem, and K. W. Bowyer. Automated poststorm damage classification of low-rise building roofing systems using high-resolution aerial imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 52(7):3851–3861, 2014. ISSN 1558-0644. doi: 10.1109/TGRS.2013.2277092.

Jim O Thomas. *Computer Vision Techniques for Damage Assessment from High Resolution Remote Sensing Imagery*. PhD thesis, University of Notre Dame, Dec 2012. URL <https://curate.nd.edu/show/rj430289f89>.

Tori Tomiczek, Andrew Kennedy, Yao Zhang, Margaret Owensby, Mark E. Hope, Ning Lin, and Abigail Flory. Hurricane damage classification methodology and fragility functions derived from hurricane sandy's effects in coastal new jersey. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 143(5):04017027, 2017. doi: 10.1061/(ASCE)WW.1943-5460.0000409. URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29WW.1943-5460.0000409>.

W. van de Lindt, Andrew Graettinger, Rakesh Gupta, Thomas Skaggs, Steven Pryor, and J. Fridley Kenneth. Performance of wood-frame structures during hurricane katrina. *Journal of Performance of Constructed Facilities*, 21(2):108–116, 2007. doi: 10.1061/(ASCE)0887-3828(2007)21:2(108). URL [https://doi.org/10.1061/\(ASCE\)0887-3828\(2007\)21:2\(108\)](https://doi.org/10.1061/(ASCE)0887-3828(2007)21:2(108)).

Anand Vetrivel, Markus Gerke, Norman Kerle, and George Vosselman. Identification of structurally damaged areas in airborne oblique images using a visual-bag-of-words approach. *Remote Sensing*, 8(3), 2016. ISSN 2072-4292. doi: 10.3390/rs8030231. URL <https://www.mdpi.com/2072-4292/8/3/231>.

Peter J. Vickery, Peter F. Skerlj, Jason Lin, Lawrence A. Twisdale, Michael A. Young, and Francis M. Lavelle. Hazus-mh hurricane model methodology. ii: Damage and loss estimation. *Natural Hazards Review*, 7(2):94–103, 2006. doi: 10.1061/(ASCE)1527-6988(2006)7:2(94). URL <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%291527-6988%282006%297%3A2%2894%29>.

Roger M. Wakimoto and Peter G. Black. Damage survey of hurricane andrew and its relationship to the eyewall. *Bulletin of the American Meteorological Society*, 75(2):189 – 202, 1994. doi: 10.1175/1520-0477(1994)075<0189:DSOHAA>2.0.CO;2. URL [https://journals.ametsoc.org/view/journals/bams/75/2/1520-0477\\_1994\\_075\\_0189\\_dsohaa\\_2\\_0\\_co\\_2.xml](https://journals.ametsoc.org/view/journals/bams/75/2/1520-0477_1994_075_0189_dsohaa_2_0_co_2.xml).

Chaofeng Wang, Qian Yu, Kincho H Law, Frank McKenna, X Yu Stella, Ertugrul Taciroglu, Adam Zsarnóczyay, Wael Elhaddad, and Barbaros Cetiner. Machine learning-based regional scale intelligent modeling of building information for natural hazard risk management. *Automation in Construction*, 122:103474, 2021. doi: <https://doi.org/10.1016/j.autcon.2020.103474>.

Charles Wang. Nheri-simcenter/surf: v0.2.0, September 2019. URL <https://doi.org/10.5281/zenodo.3463676>.

Joseph Wartman, Jeffrey W. Berman, Ann Bostrom, Scott Miles, Michael Olsen, Kurtis Gurley, Jennifer Irish, Laura Lowes, Troy Tanner, Jake Dafni, Michael Grilliot, Andrew Lyda, and Jaqueline Peltier. Research needs, challenges, and strategic approaches for natural hazards and disaster reconnaissance. *Frontiers in Built Environment*, 6:182, 2020. ISSN 2297-3362. doi: 10.3389/fbuil.2020.573068. URL <https://www.frontiersin.org/article/10.3389/fbuil.2020.573068>.

Jessica Weinkle, Chris Landsea, Douglas Collins, Rade Musulin, Ryan P. Crompton, Philip J. Klotzbach, and Roger Pielke. Normalized hurricane damage in the continental united

states 1900–2017. *Nature Sustainability*, 1(12):808–813, Dec 2018. ISSN 2398-9629. doi: 10.1038/s41893-018-0165-2. URL <https://doi.org/10.1038/s41893-018-0165-2>.

G.F. White and University of Chicago. *Human Adjustment to Floods: A Geographical Approach to the Flood Problem in the United States*. Research paper. University of Chicago, 1945. URL <https://books.google.com/books?id=cb8iAAAAMAAJ>.

Woolpert. Post event damage survey data requirements whitepaper, 2006.

Siyuan Xian, Ning Lin, and Adam Hatzikyriakou. Storm surge damage to residential areas: a quantitative analysis for hurricane sandy in comparison with fema flood map. *Natural Hazards*, 79(3):1867–1888, 2015.

Chul Min Yeum, Shirley J. Dyke, and Julio Ramirez. Visual data classification in post-event building reconnaissance. *Engineering Structures*, 155:16 – 24, 2018. ISSN 0141-0296. doi: <https://doi.org/10.1016/j.engstruct.2017.10.057>. URL <http://www.sciencedirect.com/science/article/pii/S0141029616306848>.

Min Yeum Chul, J. Dyke Shirley, Bedrich Benes, Thomas Hacker, Julio Ramirez, Alana Lund, and Santiago Pujol. Postevent reconnaissance image documentation using automated classification. *Journal of Performance of Constructed Facilities*, 33(1):04018103, 2019. doi: 10.1061/(ASCE)CF.1943-5509.0001253. URL [https://doi.org/10.1061/\(ASCE\)CF.1943-5509.0001253](https://doi.org/10.1061/(ASCE)CF.1943-5509.0001253).

Qian Yu, C. Wang, B. Cetiner, S. Yu, F. McKenna, E. Taciroğlu, and K. Law. Building information modeling and classification by visual learning at a city scale. *ArXiv*, abs/1910.06391, 2019a. doi: 10.5281/zenodo.3996808.

Qian Yu, Chaofeng Wang, Barbaros Cetiner, Stella X Yu, Frank Mckenna, Ertugrul Taciroglu, and Kincho H Law. Building information modeling and classification by visual learning at a city scale. *arXiv preprint arXiv:1910.06391*, 2019b.



Wei Zhai and Zhong-Ren Peng. Damage assessment using google street view: Evidence from hurricane michael in mexico beach, florida. *Applied Geography*, 123:102252, 2020. ISSN 0143-6228. doi: <https://doi.org/10.1016/j.apgeog.2020.102252>. URL <http://www.sciencedirect.com/science/article/pii/S0143622819303479>.

## Appendices

Appendix A

StEER Buildings - Windstorm Application Fields

Table A.1: StEER Buildings - Windstorm Application Fields

Column		Column Header	Field	Format	Response Choices/ Description
A	1	fulcrum_id	Record ID	Text	Auto-populated
B	2	status	Damage State	Single Choice	0=No Damage 1=Minor 2=Moderate 3=Severe 4=Destroyed
C	3	project	Project	Text	Windstorm Name
D	4	latitude	Latitude	Decimal	Auto-populated
E	5	longitude	Longitude	Decimal	Auto-populated
F	6	name_of_investigator	Name of Investigator	Text	
G	7	date	Date	MM/DD/YYYY	Auto-populated
H	8	general_notes	General Notes	Text	Investigator/ Librarian general notes

I	9	assessment_ type	Assessment Type	Single Choice	Aerial Drive-by On-site Remote General Area Other
J	10	all_photos	All Photos	Comma separated values	Photos associated with record
K	11	all_photos_ captions	All Photos Captions	Comma separated text	All photo captions supplied by surveyor (if any)
L	12	all_photos_ urls	Direct Path to Photo Hosted on Fulcrum	URL	Auto-populated
M	13	audio	Audio	Comma separated values	Surveyor-supplied audio
N	14	audio_url	Direct Path to Fulcrum Entry	URL	Auto-populated

O	15	overall_ damage_notes	Overall Damage Notes	Text	Overall damage notes supplied by surveyor/Librarian
P	16	hazards_present	Hazards Present	Multiple Choice (Comma separated text)	Flood Rain Surge Tree-fall Wind Wind-borne debris Unknown Other
Q	17	wind_damage_rating	Wind Damage Rating	Single Choice	-1=Not Applicable 0=No Damage 1=Minor 2=Moderate 3=Severe 4=Destroyed
R	18	surge_damage_rating	Surge Damage Rating	Single Choice	0=No Damage 1=Minor 2=Moderate 3=Severe 4=Very Severe 5=Partial Collapse 6=Collapse

S	19	rainwater_ingress_damage_rating	Rainwater Ingress Damage Rating	Single Choice	-1=Unknown 0=None Visible 1=Minor Ingress 2=Moderate 3=Severe 4=Destroyed
T	20	attribute_notes	Attribute Notes	Text	Attribute notes supplied by surveyor/Librarian
U	21	address_sub_thoroughfare	House Number	Text	Auto-populated
V	22	address_thoroughfare	Street Name	Text	Auto-populated
W	23	address_suite	Suite Number	Text	Auto-populated
X	24	address_locality	City/Town	Text	Auto-populated
Y	25	address_sub_admin_area	County	Text	Auto-populated
Z	26	address_admin_area	State	Text	Auto-populated

AA	27	address_ postal_ code	Zip Code	Text	Auto-populated
AB	28	address_ country	Country	Text	Auto-populated
AC	29	address_ full	Full Address	Text	Address supplied by surveyor/Librarian

AD	30	building_ type	Building Type	Single Choice	Single Family, Multi-Family, Apartment, Assisted Living Center , Condominium, Detached Garage, Government, Hotel/Motel, Manufactured Home, Manufacturing Plant, Marina, Office, Park Shelter, Professional, Religious, Restaurant, Retail, RV, Service Station, Shed, Supermarket, Warehouse, Unknown, Other
AE	31	number_ of_ stories	Number of Stories	Integer	1-25



AF	32	understory_ pct_of_ building_ footprint	Understory Area(% of Building Footprint)	Single Choice	0% - 100%
AG	33	first_floor_ elevation_ feet	First Floor Elevation in Feet	Decimal	0-13
AH	34	year_built	Year Built	Integer	Surveyor-supplied year built
AI	35	roof_shape	Roof Shape	Multiple Choice (Comma separated text)	Complex, Flat, Gable, Gable/Hip Combo, Gambrel, Hip, Mansard, Monoslope, Unknown, Other
AJ	36	roof_slope	Roof Slope	Integer	Surveyor-supplied roof slope
AK	37	front_ elevation_ orientation	Front Elevation Orienta- tion	Integer	Surveyor-supplied front elevation orientation
AL	38	structural_ notes	Structural Notes	Text	Structural notes from surveyor

AM	39	mwfrs	Main Wind Force Resisting System	Multiple Choice (Comma separated text)	Roof Diaphragm, wood, Roof Diaphragm, steel, Roof Diaphragm, concrete, Roof Diaphragm, composite, Wall Diaphragm, wood, Wall Diaphragm, steel, Wall Diaphragm, concrete, Wall Diaphragm, masonry, Wall, X-bracing, Moment Frame, Unknown, Other
----	----	-------	--	--	---

AN	40	foundation_ type	Foundation Type	Multiple Choice (Comma separated text)	Slab-on-grade, Cast-in-place concrete piers, Ground anchors and strapping, Crawlspace, Reinforced masonry piers, Reinforced masonry stem wall, Unreinforced masonry piers, Unreinforced masonry stem wall, Wood Piers <= 8 ft, Wood Piers >8 ft, Unknown , Other
AO	41	wall_ anchorage_ type	Wall Anchorage Type	Multiple Choice (Comma separated text)	Anchor bolts with nuts and washers, Anchor bolts with missing nuts and washers, Metal straps, Concrete nails, Unknown, Other

AP	42	wall_ structure	Wall Structure	Multiple Choice (Comma separated text)	Wood frame, Masonry (reinforced), Masonry (unreinforced), Masonry (unknown), Concrete, tilt-up, Concrete, moment resisting frame, Steel, moment resisting frame, Steel, braced frame, Steel, cold form, Insulated concrete form (ICF) walls, Solid Brick Wythe , Unknown, Other
----	----	--------------------	-------------------	--	---

AQ	43	wall_ substrate	Wall Substrate	Multiple Choice (Comma separated text)	Wood, sheathing (continuous), Wood, sheathing (corners only), Wood,dimensional planks, Insulated sheathing, Insulated foam board, Non-engineered wood panel, Metal panels, Not Applicable, Unknown, Other
----	----	--------------------	-------------------	--	--

AR	44	wall_cladding	Wall Cladding	Multiple Choice (Comma separated text)	Aluminum siding, Brick, Curtain wall, EIFS, FiberCement Board, Corrugated steel panels, Plywood Siding, Stucco, Vinyl Siding (standard), Vinyl Siding (high wind rated), Vinyl Siding (unknown), Wood Boards, Wood Shake/Shingle, Unknown, Other
AS	45	soffit_type	Soffit Type	Multiple Choice (Comma separated text)	None, Vinyl, Metal, Wood, Unknown, Other
AT	46	front_wall_fenestration_ratio	Front Wall Fenestration Ratio	Single Choice	0%-100%

AU	47	front_ wall_ fenestration_ protection	Front Wall Fenestration Protection	Multiple Choice (Comma separated text)	None, Unknown, Impact Resistant, Plywood/OSB Panel, Hurricane Shutter, Other
AV	48	left_ wall_ fenestration_ ratio	Left Wall Fenestration Ratio	Single Choice	0%-100%
AW	49	left_ wall_ fenestration_ protection	Left Wall Fenestration Protection	Multiple Choice (Comma separated text)	None, Unknown, Impact Resistant, Plywood/OSB Panel, Hurricane Shutter, Other
AX	50	back_ wall_ fenestration_ ratio	Back Wall Fenestration Ratio	Single Choice	0%-100%
AY	51	back_ wall_ fenestration_ protection	Back Wall Fenestration Protection	Multiple Choice (Comma separated text)	None, Unknown, Impact Resistant, Plywood/OSB Panel, Hurricane Shutter, Other

AZ	52	right_ wall_ fen- estration_ ratio	Right Wall Fenestra- tion Ratio	Single Choice	0%-100%
BA	53	right_ wall_ fen- estration_ protection	Right Wall Fenestra- tion Protection	Multiple Choice (Comma separated text)	None, Unknown, Impact Resistant, Plywood/OSB Panel, Hurricane Shutter, Other
BB	54	large_ door_ present	Large Door Present	Multiple Choice (Comma separated text)	Yes, No, N/A



BC	55	large_ door_ opening_ type_ front	Large Door Opening Type Front	Multiple Choice (Comma separated text)	None, Single garage door (standard), Double garage door (standard), Single garage door (wind-rated), Double garage door (wind-rated), Single garage door (unknown), Double garage door (unknown), Sectional door, Roll-up door, Other
----	----	--	---	--	--

BD	56	large_ door_ opening_ type_ left	Large Door Opening Type Left	Multiple Choice (Comma separated text)	None, Single garage door (standard), Double garage door (standard), Single garage door (wind-rated), Double garage door (wind-rated), Single garage door (unknown), Double garage door (unknown), Sectional door, Roll-up door, Other
----	----	---	---------------------------------------	--	--

BE	57	large_ door_ opening_ type_ back	Large Door Opening Type Back	Multiple Choice (Comma separated text)	None, Single garage door (standard), Double garage door (standard), Single garage door (wind-rated), Double garage door (wind-rated), Single garage door (unknown), Double garage door (unknown), Sectional door, Roll-up door, Other
----	----	---	---------------------------------------	--	--

BF	58	large_ door_ opening_ type_ right	Large Door Opening Type Right	Multiple Choice (Comma separated text)	None, Single garage door (standard), Double garage door (standard), Single garage door (wind-rated), Double garage door (wind-rated), Single garage door (unknown), Double garage door (unknown), Sectional door, Roll-up door, Other
BG	59	roof_system	Roof System	Multiple Choice (Comma separated text)	Steel, cold formed, Steel, hot rolled, Steel, joists, Concrete slab, Wood, rafter, Wood, trusses, Wood, unknown, Unknown, Other

BH	60	r2wall_ at- tachment	Roof to Wall At- tachment	Multiple Choice (Comma separated text)	Toe-nails, Metal ties, Metal straps, Bolted connection, Welded connection, Unknown, Other
BI	61	r2w_ at- tachment_ type	Roof to Wall At- tachment Type	Text	Surveyor-supplied roof to wall attachment type
BJ	62	roof_ substrate_ type	Roof Substrate Type	Multiple Choice (Comma separated text)	Plywood/OSB, Dimensional lumber, Metal deck, Concrete, None, Unknown, Other

BK	63	roof_cover	Roof Cover	Multiple Choice (Comma separated text)	Asphalt shingles (laminated), Built-up with Gravel, Built-up without Gravel, Clay tiles, Concrete tiles, Metal shingles, Metal, corrugated, Metal, standing seam, Roll roofing, Single ply, Wood shake, Wood shingle, Unknown, Other
BL	64	secondary_ water_ barrier	Secondary Water Barrier	Multiple Choice (Comma separated text)	None, Closed-cell urethane foam adhesive, Fully adhered membrane, High performance underlayment, Self-adhering membrane over joints, Unknown, Other

BM	65	overhang_ length	Overhang Length	Integer	Surveyor-supplied overhang length
BN	66	parapet_ height_ inches	Parapet Height in inches	Integer	Surveyor-supplied parapet height
BO	67	wind_ damage_ details	Wind Damage Details	Text	Wind damage notes from surveyor
BP	68	roof_ structure_ damage_	Roof Structure Damage	Single Choice	0%-100%
BQ	69	roof_ substrate_ damage	Roof Substrate Damage	Single Choice	0%-100%
BR	70	roof_ cover_ damage_	Roof Cover Damage	Single Choice	0%-100%
BS	71	wall_ structure_ damage_	Wall Structure Damage	Single Choice	0%-100%
BT	72	wall_s ubstrate_ damage_	Wall Substrate Damage	Single Choice	0%-100%
BU	73	building_ envelope_ damage_	Building Envelope Damage	Single Choice	0%-100%

BV	74	front_ wall_ fen- estration_ damage	Front Wall Fenestra- tion Damage	Single Choice	0%-100%
BW	75	left_ wall_ fenestra- tion_ damage	Left Wall Fenestra- tion Damage	Single Choice	0%-100%
BX	76	back_ wall_ fenestra- tion_ damage	Back Wall Fenestra- tion Damage	Single Choice	0%-100%
BY	77	right_ wall_ fen- estration_ damage	Right Wall Fenestra- tion Damage	Single Choice	0%-100%
BZ	78	large_ door_ failure	Large Door Failure	Multiple Choice (Comma separated text)	None, Front, Left, Back, Right, All, other
CA	79	soffit_damage	Soffit Damage	Single Choice	0%-100%
CB	80	fascia_damage	Fascia Damage	Single Choice	0%-100%



CC	81	stories_ with_ damage	Stories with Damage	Integer	Surveyor-supplied stories with damage
CD	82	water_ induced_ damage_ notes	Water Induced Damage Notes	Text	Water induced damage notes from surveyor
CE	83	percent_ of_ building_ footprint_ eroded	Percent of Building Footprint Eroded	Single Choice	0%-100%
CF	84	__damage_ to_ understory	Damage to Under- story	Single Choice	0%-100%
CG	85	maximum_ scour_ depth_ inches	Maximum Scour Depth in inches	Integer	Surveyor-supplied maximum scour depth
CH	86	__piles_ missing_ or_ collapsed	Piles Missing or Collapsed	Single Choice	0%-100%
CI	87	-- piles_ leaning_ or_ broken	Piles Leaning or Broken	Single Choice	0%-100%

CJ	88	cause_of_foundation_damage	Cause of Foundation Damage	Multiple Choice (Comma separated text)	Erosion, Wave, Flood, Floating Debris, Velocity Scour, None, Unknown, Other
CK	89	reroof_year	Reroof Year	Integer	Surveyor-supplied reroof year
CL	90	retrofit_type_1	Retrofit Type 1	Text	Surveyor-supplied retrofit description
CM	91	retrofit_1_year	Retrofit 1 Year	Integer	Surveyor-supplied retrofit year
CN	92	retrofit_type_2	Retrofit Type 2	Text	Surveyor-supplied retrofit description
CO	93	retrofit_2_year	Retrofit 2 year	Integer	Surveyor-supplied retrofit year
CP	94	data_librarians	Data Librarian	Text	Data Librarian Name
CQ	95	qc_progress_code	QC Progress Code	Single Choice	1, 1, 2, 2e, 3, 3e
CR	96	qc_notes	QC Notes	Text	Notes from Data Librarians regarding the DE/QC process

Appendix B  
Automation Framework Code

Listing B.1: Libraries used

---

```
#Importing all necessary libraries
%matplotlib inline
import urllib
import requests
from bs4 import BeautifulSoup
import re
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait as wait
from configure import *
from skimage import io
from PIL import Image
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
import pprint
pp = pprint.PrettyPrinter(indent=4)
from skimage import color, img_as_float
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from skimage.feature import hog
```

```

from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_predict
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import skimage
from skimage.io import imread
from joblib import dump, load
import json
from shapely.geometry import shape, Point
import random
import geopandas as gpd
from multiprocessing.dummy import Pool as ThreadPool
from brails.modules import RoofClassifier
from brails.modules import OccupancyClassifier
from brails.modules import SoftstoryClassifier
from brails.modules import YearBuiltClassifier
from brails.modules import NFloorDetector
from surf.NN import SpatialNeuralNet

```

---

Listing B.2: Source code from BRAILS to download images from Google API

---

```

# THIS CODE IS COPIED FROM BRAILS FILES (downloadRoofImages.py)
# These functions are called later on in the
# notebook to download the roof images
# and Street View images using Google API
def download(urls):
    i = -1
    for ls in urls:
        i +=1
        urlTop = ls[0]
        urlStreet = ls[1]
        lon = ls[2]

```

```

lat = ls[3]

thisFileDir = roofDownloadDir
if not os.path.exists(thisFileDir):
    os.makedirs(thisFileDir)
numoffiles = len(os.listdir(thisFileDir))
if numoffiles < maxNumofRoofImgs:
    picname = thisFileDir + '/{}.png'.format(i)
    if not os.path.exists(picname):
        r = requests.get(urlTop)
        f = open(picname, 'wb')
        f.write(r.content)
        f.close()
    else:
        break
thisFileDir = streetDownloadDir
if not os.path.exists(thisFileDir):
    os.makedirs(thisFileDir)
numoffiles = len(os.listdir(thisFileDir))
if numoffiles < maxNumofRoofImgs:
    picname = thisFileDir + '/{}.png'.format(i)
    if not os.path.exists(picname):
        r = requests.get(urlStreet)
        f = open(picname, 'wb')
        f.write(r.content)
        f.close()
    else:
        break

# This function verifies that the
# coordinates obtained from Google fall within the
# building footprints
def getBIMIndex(pts):

```

```

poly = Polygon(pts)
o = pts[0]
distance , index = kdTree.query(o,10) # nearest 10 points
trueIndex = []
[trueIndex.append(i) for i in index if Point(coordsAll[i]).within(poly)]

if trueIndex:
    theInd = trueIndex[0]

else:# no point inside polygon
    theInd = index[0]

return theInd

```

---

### Listing B.3: Input processing module

---

```

# Importing the provided addresses or coordinates
# Input 1 if input is list of addresses
# Input 2 if input is list of coordinates

choice = input('Are you providing a list of addresses (enter: 1)
or coordinates (enter:2)? ')

if int(choice) == 1:
    filename = input('Enter csv file name with addresses. Do not include the.
csv extension in the name you enter. ')
    addresses = pd.read_csv
('inputs/{}.csv'.format(filename))
    addresses = addresses['address'].values

elif int(choice) ==2:

```

```

filename = input('''Enter csv file name with coordinates.
Do not include the .csv extension in the name you enter.''')
df = pd.read_csv('inputs/{}.csv'.format(filename))
lats = df['lat'].values
lons = df['long'].values

addresses = []

df['geocode_data'] = ''
# Reverse geocoding to find addresses from coordinates
# Used in web scraping
def reverseGeocode(latlng):
    result = {}
    apikey = 'AIzaSyBohpHY6irneiV00vnEF1o1yI8luejbdII'
    url = '''https://maps.googleapis.com/
maps/api/geocode/json?latlng={latlng}&key={apikey}'''
    r = requests.get(url)
    r.raise_for_status()
    data = r.json()
    if data['results']:
        result = data['results'][0]
    return result

for i, row in df.iterrows():
    df['geocode_data'][i] = reverseGeocode
    (df['lat'][i].astype(str) +
    ', ' + df['long'][i].astype(str))
for i, row in df.iterrows():
    address = row['geocode_data']['formatted_address']
    address = address.replace(', ', ',')
    addresses.append(address)

```

```
else:
    print('Please enter 1 for addresses or 2 for coordinates')
```

---

#### Listing B.4: Web scraping module

---

```
# Web scraping for basic building attributes including :
# year built, number of stories,
# foundation type, and occupancy
def web_scraping(addresses, attr):
    driver = webdriver.Chrome('chromedriver')
    found = 0
    #finding year built
    if attr == 'year_built':
        years = []
        for i in range(len(addresses)):
            time.sleep(2)
            print(''Web scraping year built
information for address: {}/{}... '''.format
                (i+1,len(addresses)), end='\r')

            if i % 50 == 0:
                driver.close()
                driver = webdriver.Chrome('chromedriver')
            query = addresses[i] + '_built_in'
            query = query.replace('_', '+')
            URL = f"https://google.com/search?q={query}"
            driver.get(URL)
            # div.IsZvec was obtained from using inspect
            # element feature in Google Chrome
            snippets = wait(driver, 60).until
                (lambda driver: driver.find_elements_by_css_selector("div.IsZvec"))
            for j in range(len(snippets)):
                # Lower case to make searching for term easier
```



```

        snippet = snippets[j].text.lower()
        year_built = 'NA'
        if 'built' in snippet:
            index = snippet.find("built")
            # Obtaining the number within
            # the string that we are interested in
            #and making sure it is a year built >1000
            string = snippet[index+4:index+15]
            test = ''.join(filter(lambda s: s.isdigit(), string))
            if test > '1000':
                year_built = test
                found = found+1
                break
        years.append(year_built)
    driver.close()
    print(''Finished processing {} addresses.
    Number of found year built data: {}''
        .format(len(addresses),found))
    return years
#finding number of stories
elif attr == 'no_stories':
    stories = []
    for i in range(len(addresses)):
        time.sleep(2)
        print(''Web scraping number of
        stories information for address: {}/{}... ''
            .format(i+1,len(addresses)), end='\r')
        if i % 50 == 0:
            driver.close()
            driver = webdriver.Chrome('chromedriver')
            query = addresses[i] + '_stories'
            query = query.replace('_', '+')
            URL = f"https://google.com/search?q={query}"

```

```

driver.get(URL)
snippets = wait(driver, 60).until
(lambda driver:driver.find_elements_by_css_selector("div.IsZvec"))
for j in range(len(snippets)):
    snippet = snippets[j].text.lower()
    no_stories = 'NA'
    if 'stories' in snippet:
        index = snippet.find("stories")
        string = snippet[index+6:index+10]
        test = ''.join(filter(lambda s: s.isdigit(), string))
        if test > '0':
            no_stories = test
            found = found+1
            break
    stories.append(no_stories)
driver.close()
print(''Finished processing {} addresses.
Number of stories data found: {}'')
    .format(len(addresses),found))
return stories
#finding foundation type
elif attr == 'foundation_type':
    foundations = []
    # Used the most common foundation types found from our data
    ftypes = ['slab', 'concrete_piers', 'wood_piers', 'stem_wall']
    for i in range(len(addresses)):
        time.sleep(2)
        print(''Web scraping foundation
type for address: {}/{}...'')
            .format(i+1,len(addresses)), end='\r')
    if i % 50 == 0:
        driver.close()
        driver = webdriver.Chrome('chromedriver')

```

```

query = addresses[i] + '_foundation_type'
query = query.replace('_', '+')
URL = f"https://google.com/search?q={query}"
driver.get(URL)
snippets = wait(driver, 60).until
(lambda driver: driver.find_elements_by_css_selector("div.IsZvec"))
for j in range(len(snippets)):
    snippet = snippets[j].text.lower()
    foundation_type = 'NA'
    for ftype in ftypes:
        if ftype in snippet:
            foundation_type = ftype
            found = found+1
            break
    if foundation_type != 'NA':
        break
    foundations.append(foundation_type)
driver.close()
print('''Finished processing {} addresses.
Number of foundation type data found: {}'''.
      .format(len(addresses), found))
return foundations
#occupancy type
elif attr == 'occupancy_type':
    occupancy = []
    # Using the most common occupancy types to search for
    occTypes =
    ['single_family', 'multi_family', 'business', 'professional', 'apartment']
    for i in range(len(addresses)):
        time.sleep(2)
        print('Web_scraping_occupancy_type_for_address:_{}/{}...'.
              .format(i+1, len(addresses)), end='\r')
        if i % 50 == 0:

```

```

        driver.close()
        driver = webdriver.Chrome('chromedriver')
        query = addresses[i] + '_type'
        query = query.replace('_', '+')
        URL = f"https://google.com/search?q={query}"
        driver.get(URL)
        snippets = wait(driver, 60).until
        (lambda driver: driver.find_elements_by_css_selector("div.IsZvec"))
        for j in range(len(snippets)):
            snippet = snippets[j].text.lower()
            occupancy_type = 'NA'
            for occType in occTypes:
                if occType in snippet:
                    occupancy_type = occType
                    found = found+1
                    break
            if occupancy_type != 'NA':
                break
            occupancy.append(occupancy_type)
        driver.close()
        print(f'''Finished processing {
addresses. Number of occupancy type data found: {
                .format(len(addresses), found)
            return occupancy

elif attr == 'roof_shape':
    rshapes = []
    # Used the most common roof types found from our data
    rtypes = ['gable', 'hip', 'flat']
    for i in range(len(addresses)):
        time.sleep(2)
        print(f'''Web scraping roof shape for address: {
                .format(i+1, len(addresses)), end='\r')

```

```

if i % 50 == 0:
    driver.close()
    driver = webdriver.Chrome('chromedriver')
    query = addresses[i] + '_roof_shape'
    query = query.replace('_', '+')
    URL = f"https://google.com/search?q={query}"
    driver.get(URL)
    snippets = wait(driver, 60).until
    (lambda driver: driver.find_elements_by_css_selector("div.IsZvec"))
    for j in range(len(snippets)):
        snippet = snippets[j].text.lower()
        rshape = 'NA'
        for rtype in rtypes:
            if rtype in snippet:
                rshape = rtype
                found = found+1
                break
        if rshape != 'NA':
            break
        rshapes.append(rshape)
    driver.close()
    print( '''Finished processing {} addresses.
    Number of roof shapes found: {} '''
        .format(len(addresses), found))
    return rshapes

else:
    print( 'Please provide valid input ' )

```

---

Listing B.5: Calling the web scraping module and saving results

---

```

# Calling the functions above to perform web scraping
years = web_scraping(addresses, 'year_built')

```

```

stories = web_scraping(addresses , 'no_stories ')
foundations = web_scraping(addresses , 'foundation_type ')
occupancy = web_scraping(addresses , 'occupancy_type ')
roofs = web_scraping(addresses , 'roof_shape ')

# Saving web scraping results to be used in the next steps
df = list(zip(addresses , stories , years , occupancy , foundations , roofs))
Frame=pd.DataFrame(df, columns =
                    ["address" , "stories" ,"years" , "occupancy" , "foundations" ,"roofs" ])
Frame.to_csv( "Outputs\Dataset_Results\web_scraping_results.csv"
             , index=False , encoding='utf-8-sig ')

```

---

Listing B.6: Obtaining and saving coordinates from addresses module

---

```

# Saving Google maps GeoJSON file
# to get the coordinates for each
# address from it/ this uses Google API key
geocode_dec = 'no'
if int(choice) == 2:
    while True:

        geocode_dec = input( '''You have already provided coordinates ,
        would you like to use Google API to download geoJSON files
        to extract coordinates from them? Answer "yes or "no". ''' )

        if geocode_dec == 'yes' or geocode_dec == 'no':
            break
        else:
            print( '''Please enter either "yes"
            or "no" without quotation marks. ''' )

if int(choice) == 1 or geocode_dec == 'yes':
    baseurl_addr_google= '''https://maps.googleapis

```

```

.com/maps/api/geocode/json?address={}&key= ''
+GoogleMapAPIKey
baseurl_addr = baseurl_addr_google
for addr in addresses:
    url = baseurl_addr.format(addr)
    url.replace('_', '+')
    r = requests.get(url)
    f = open('Outputs/Geocoding/geocode_{}.geojson'.
            format(addr), 'wb')
    f.write(r.content)
    f.close()

# Saving the coordinates for each address from the
# geojson files we got from Google maps in the previous step

if int(choice) == 1 or geocode_dec == 'yes':
    lats = []
    lons = []
    addrs = []

for addr in addresses:
    with open('Outputs/Geocoding/geocode_{}.geojson'.
            format(addr)) as f:
        js = json.load(f)
        if js['status'] == 'OK':
            d = js['results'][0]
            coord = d['geometry']['location']
            lat = coord['lat']
            lon = coord['lng']
            lats.append(lat)
            lons.append(lon)
            addrs.append(addr)

```

```

    # Saving the coordinates in a csv file
df = list(zip(addresses , lats , lons))
Frame = pd.DataFrame(df, columns = ["address", "lat", "lon"])
Frame.to_csv( "Outputs/Dataset_Results/Addr_Coords.csv", index=False ,
              encoding='utf-8-sig' )

df = pd.read_csv( 'inputs/Framework_Test_Data.csv' .format(filename))
lats = df['lat'].values
lons = df['lon'].values
addrs = df['address'].values
df = list(zip(addresses , lats , lons))
Frame = pd.DataFrame(df, columns = ["address", "lat", "lon"])
Frame.to_csv( "Outputs/Dataset_Results/Addr_Coords.csv",
              index=False , encoding='utf-8-sig' )

```

---

#### Listing B.7: Validating coordinates using building footprints module

---

```

# Depending on whether you have building footprints file
#or you want to use building footprints from Microsoft AI generated
#or you don't want to use any
# at all the code will give the same result
# The building footprints are used to check
# if the coordinates fall within the building footprints

fp_choice = input( '''Please enter your footprints
choice as follows: "myBuildingFPs" or "state" or "noFPs"''' )

# This runs if you have
# a building footprints file for your coordinates
#The myBuildingFPs option uses source code from BRAILS
if fp_choice == 'myBuildingFPs':
    with open(cleanedBIMFile_w_coord_Name, 'r') as addrfile ,
        open(resultBIMFileName, 'w+', newline='') as resultBIMFile ,

```



```
open(BuildingFootPrintsFileName) as BuildingFootPrintsFile:
```

```
    addrcsv = list(csv.reader(addrfile))
    colNames = addrcsv[0][0:]
    addrcsv = addrcsv[1:]
    coordsAll = []
    [coordsAll.append([row[2], row[1]]) for row in addrcsv]
    coordsAll = np.array(coordsAll, dtype=np.float32)
    kdTree = spatial.KDTree(coordsAll)
```

```
    fps = []
    bldgFootPrints = json.load(BuildingFootPrintsFile)
    bldgFootPrintsFeatures = bldgFootPrints["features"]
    bldgFootPrintsFeatures = bldgFootPrintsFeatures[0:]
    features = []
```

```
for bfpf in bldgFootPrintsFeatures:
    pts = bfpf['geometry']['coordinates'][0]
    bimIndex = getBIMIndex(pts)
    info = addrcsv[bimIndex]
    for i in range(len(colNames)):
        vName = colNames[i]
        if info[i] == 'None' or info[i] == '':
            bfpf['properties'][colNames[i]] = None
            continue
        elif vName in intVariables:
            vValue = int(float(info[i]))
        elif vName in floatVariables:
            vValue = float(info[i])
        else:
            vValue = info[i]
        bfpf['properties'][colNames[i]] = vValue
    features.append(bfpf)
```

```

    bldgFootPrints[ 'features' ] = features
    json.dump(bldgFootPrints, resultBIMFile)
    print("bim_has_been_added_to_{}".format(resultBIMFileName))

outputDir = roofDownloadDir
baseurl_streetview = '''https://maps.googleapis.
com/maps/api/streetview?
size=200x200&location={lat},{lon}&fov=80&pitch=0&key='''
+GoogleMapAPIKey
baseurl_satellite= '''https://maps.googleapis.com/maps
/api/staticmap?center={lat},
{lon}&zoom=20&scale=1&size=256x256&
maptype=satellite&key='''+
GoogleMapAPIKey+ '''&format=png&visual_refresh=true'''

cityFile = gpd.read_file(resultBIMFileName).to_json()
footjsons = json.loads(cityFile)[ 'features' ]
urls = []
for j in footjsons:
    address = j[ 'properties' ][ 'address' ]
    lat = j[ 'properties' ][ 'lat' ]
    lon = j[ 'properties' ][ 'lon' ]

    urlTop = baseurl_satellite.format(lat=lat, lon=lon)
    urlStreet = baseurl_streetview.format(lat=lat, lon=lon)
    urls.append([urlTop, urlStreet, lon, lat])
print('shuffling ... ')
random.shuffle(urls)
print('shuffled ... ')

ncpu = 1
step = int(len(urls)/ncpu)+1
chunks = [urls[x:x+step] for x in range(0, len(urls), step)]

```

```

print( 'Downloading satellite images of roofs from Google API... ')
# get some workers
pool = ThreadPool(ncpu)
# send job to workers
results = pool.map(download, chunks)
# jobs are done, clean the site
pool.close()
pool.join()
print( 'Satellite images of roofs downloaded... ')

=====
# This runs if you want to use the Microsoft
# building footprints generated by AI
#It uses the file for the entire state
elif fp_choice == 'state':

    # load GeoJSON file containing sectors
    with open('Inputs/Footprints.geojson') as f:
        js = json.load(f)

    # construct point based on lon/lat returned by geocoder
    addr_coords = pd.read_csv
    ('Outputs/Dataset_Results/Addr_Coords.csv')
    lats = addr_coords['lat'].values
    lons = addr_coords['lon'].values
    k = 0
    points = []
    correct_points = []
    bad_points = []
    for x in range(len(addr_coords)):
        point = Point(lons[k], lats[k])
        points.append(point)

```

```

    k = k+1
# check each polygon to see if it contains the point
i=0
j= len(js['features'])
for feature in js['features']:
    print('checking_{}/{}'.format(i, j), end='\r')
    i = i+1
    polygon = shape(feature['geometry'])
    for point in points:
        if polygon.contains(point):
            correct_points.append(point)
            break

for point in points:
    if point not in correct_points:
        bad_points.append(point)

correct_lons = []
correct_lats = []
bad_lons = []
bad_lats = []
for point in correct_points:
    correct_lons.append(point.x)
    correct_lats.append(point.y)
for point in bad_points:
    bad_lons.append(point.x)
    bad_lats.append(point.y)

df = list(zip(bad_lats, bad_lons))
Frame = pd.DataFrame(df, columns =
                    ["bad_lat", "bad_lon"])
Frame.to_csv
( "Outputs/Dataset_Results/Bad_Coords.csv",

```

```

index=False , encoding='utf-8-sig')

#copied from noFPs below for now:
outputDir = roofDownloadDir
baseurl_streetview = '''https://maps.googleapis.
com/maps/api/streetview?
size=200x200&location={lat},{lon}&fov=80&pitch=0&key='''
+GoogleMapAPIKey
baseurl_satellite= '''https://maps.googleapis.
com/maps/api/staticmap?center=
{lat},{lon}&zoom=20&scale=1&size=256x256
&maptype=satellite&key='''
+GoogleMapAPIKey+ '''&format=png&visual_refresh=true'''
i=0
urls = []
for lat in lats:
    urlTop = baseurl_satellite.format(lat=lat ,lon=lons[i])
    urlStreet = baseurl_streetview.format(lat=lat ,lon=lons[i])
    urls.append([urlTop , urlStreet , lons[i] , lat])
    i += 1

ncpu = 1
step = int(len(urls)/ncpu)+1
chunks = [urls[x:x+step] for x in range(0, len(urls), step)]
print('Downloading satellite images of roofs from Google API...')
# get some workers
pool = ThreadPool(ncpu)
# send job to workers
results = pool.map(download, chunks)
# jobs are done, clean the site
pool.close()
pool.join()
print('Satellite images of roofs downloaded...')

```

```

#=====  

# This runs if you don't have  

# building footprints and do not want to provide any  

elif fp_choice == 'noFPs':  

# THIS CODE IS COPIED/ADAPTED FROM #  

# BRAILS FILES (downloadRoofImages.py) #  

    outputDir = roofDownloadDir  

    baseurl_streetview = '''https://maps.googleapis.  

com/maps  

/api/streetview?  

size=200x200&location={lat},{lon}&fov  

=80&pitch=0&key='''+GoogleMapAPIKey  

baseurl_satellite= '''https://maps.googleapis.com/maps  

/api/staticmap?center=  

{lat},{lon}&zoom=20&scale=1&size=256x256&maptype=  

satellite&key='''+  

GoogleMapAPIKey+'&format=png&visual_refresh=true'  

    urls = []  

#     for lat in lats:  

while i < len(addresses):  

        urlTop = baseurl_satellite.format(lat=lats[i],lon=lons[i])  

        urlStreet = baseurl_streetview.format(lat=lats[i],lon=lons[i])  

        urls.append([urlTop, urlStreet, lons[i], lats[i]])  

        i += 1  

ncpu = 1  

step = int(len(urls)/ncpu)+1  

chunks = [urls[x:x+step] for x in range(0, len(urls), step)]  

print('Downloading_satellite_images_of_roofs_from_Google_API...')  

# get some workers  

pool = ThreadPool(ncpu)  

# send job to workers

```

```

results = pool.map(download, chunks)
# jobs are done, clean the site
pool.close()
pool.join()
print('Satellite_images_of_roofs_downloaded...')

else:
    print('Please_provide_your_footprints_choice')

```

---

Listing B.8: Downloading, cropping, and saving NOAA aerial imagery

---

```

# Downloading NOAA images
coord = pd.read_csv(''Outputs/Dataset-Results/
  Addr_Coords.csv'')
latitudes = coord['lat'].values
longitudes = coord['lon'].values
driver = webdriver.Chrome('chromedriver')
driver.set_window_size(500,500)
for i in range(len(latitudes)):
    driver.get(''https://storms.ngs.noaa.gov/storms/{}/
  index.html#20/{}/{}'')
        .format(storm, latitudes[i], longitudes[i]))
    if i < 3:
        zoom = driver.find_element_by_xpath
        (''//*[@id="map"]/div[2]/div[1]/div[1]/a[1]'')
        zoom.click()
        zoom.click()
        zoom.click()
        driver.get(''https://storms.ngs.noaa.gov/storms/{}/
  /index.html#20/{}/{}'')
            .format(storm, latitudes[i], longitudes[i]))
time.sleep(2)
screenshot = driver.save_screenshot(''Outputs/Collected-Images

```

```

        /Post_Windstorm_Roofs/temp/{}.png'''.format(i))
driver.close()

# Cropping NOAA Images

# Setting the points for cropped image from top left corner
topleft_X = 150
topleft_Y = 83
bottomright_X = 350
bottomright_Y = 283
# Opening and cropping the images to 200x200 image size
for i in range(len(latitudes)):
    im = Image.open(''Outputs/Collected_Images/
        Post_Windstorm_Roofs/temp/{}.png'''.format(i))
    cropped = im.crop
        ((topleft_X, topleft_Y, bottomright_X, bottomright_Y))
    cropped.save(''Outputs/Collected_Images/
        Post_Windstorm_Roofs/{}.png'''.format(i))
# Deleting uncropped images
files = glob.glob
('Outputs/Collected_Images/Post_Windstorm_Roofs/temp/*')
for f in files:
    os.remove(f)

```

---

### Listing B.9: Calling BRAILS classifiers

---

```

# Calling BRAILS classifiers

# initialize roof classifier
roofModel = RoofClassifier()

# initialize occupancy classifier
occupancyModel = OccupancyClassifier()

```



```

# initialize year build classifier
yearModel = YearBuiltClassifier()

# initialize no. of stories classifier
nfloorDetector = NFloorDetector()

roof_imgs = []
streetView_imgs = []
for i in range(len(addresses)):
    roof_imgs.append
    ('Outputs/Collected_Images/BRAILS_Roofs/{}.png'.format(i))
    streetView_imgs.append
    ('Outputs/Collected_Images/BRAILS_StreetView/{}.png'.format(i))

predictions = roofModel.predict(roof_imgs)

# use the occupancy classifier

predictions = occupancyModel.predict(streetView_imgs)

# use the year built classifier

predictions = yearModel.predict(streetView_imgs)

# use no. of stories classifier

predictions = nfloorDetector.predict(streetView_imgs)

occ_pred = pd.read_csv('tmp/occupancy_preds.csv')
occ_pred = occ_pred['prediction'].values
roof_pred = pd.read_csv('tmp/roofType_preds.csv')
roof_pred = roof_pred['prediction'].values

```

```

yb_pred = pd.read_csv('tmp/YearBuilt.csv')
yb_pred = yb_pred['prediction'].values
no_floors_pred = pd.read_csv('nFloorPredict.csv')
no_floors_pred = no_floors_pred['_nFloors'].values

files = glob.glob('*.*.csv')
for f in files:
    os.remove(f)
files = glob.glob('debug.log')
for f in files:
    os.remove(f)

```

---

#### Listing B.10: Using SURF on web scraped results

---

*# Here SURF is trained on the web scraped data:*

```

years_data = []
stories_data = []
foundations_data = []
occupancy_data = []
roofs_data = []

years_missing = []
stories_missing = []
foundations_missing = []
occupancy_missing = []
roofs_missing = []

for i in range(len(addresses)):
    if years[i] != 'NA':
        years_data.append([lons[i], lats[i], int(years[i])])
    elif years[i] == 'NA':
        years_missing.append([lons[i], lats[i]])

```

```

if stories[i] != 'NA':
    stories_data.append([lons[i], lats[i], int(stories[i])])
elif stories[i] == 'NA':
    stories_missing.append([lons[i], lats[i]])

if foundations[i] != 'NA':
    foundations_data.append([lons[i], lats[i], foundations[i]])
elif foundations[i] == 'NA':
    foundations_missing.append([lons[i], lats[i]])

if occupancy[i] != 'NA':
    occupancy_data.append([lons[i], lats[i], occupancy[i]])
elif occupancy[i] == 'NA':
    occupancy_missing.append([lons[i], lats[i]])

if roofs[i] != 'NA':
    roofs_data.append([lons[i], lats[i], roofs[i]])
elif roofs[i] == 'NA':
    roofs_missing.append([lons[i], lats[i]])

foundation_data_rep = []
for data in foundations_data:
    if data[2] == 'slab':
        foundation_data_rep.append([data[0], data[1], 0])
    elif data[2] == 'concrete_piers':
        foundation_data_rep.append([data[0], data[1], 1])
    elif data[2] == 'wood_piers':
        foundation_data_rep.append([data[0], data[1], 2])
    elif data[2] == 'stem_wall':
        foundation_data_rep.append([data[0], data[1], 3])

occupancy_data_rep = []

```

```

for data in occupancy_data:
    if data[2] == 'single_family':
        occupancy_data_rep.append([data[0], data[1], 0])
    elif data[2] == 'multi_family':
        occupancy_data_rep.append([data[0], data[1], 1])
    elif data[2] == 'business':
        occupancy_data_rep.append([data[0], data[1], 2])
    elif data[2] == 'professional':
        occupancy_data_rep.append([data[0], data[1], 3])
    elif data[2] == 'apartment':
        occupancy_data_rep.append([data[0], data[1], 4])

```

```

roofs_data_rep = []

```

```

for data in roofs_data:
    if data[2] == 'gable':
        roofs_data_rep.append([data[0], data[1], 0])
    elif data[2] == 'hip':
        roofs_data_rep.append([data[0], data[1], 1])
    elif data[2] == 'flat':
        roofs_data_rep.append([data[0], data[1], 2])

```

```

years_data = np.array(years_data)
stories_data = np.array(stories_data)
foundation_data_rep = np.array(foundation_data_rep)
occupancy_data_rep = np.array(occupancy_data_rep)
roofs_data_rep = np.array(roofs_data_rep)

```

```

years_missing = np.array(years_missing)
stories_missing = np.array(stories_missing)
foundations_missing = np.array(foundations_missing)
occupancy_missing = np.array(occupancy_missing)
roofs_missing = np.array(roofs_missing)

```

```

nn_years = SpatialNeuralNet(rawData = years_data , numNei = 5)
nn_stories = SpatialNeuralNet(rawData = stories_data , numNei = 5)
nn_foundations = SpatialNeuralNet(rawData = foundation_data_rep , numNei = 5)
nn_occupancy = SpatialNeuralNet(rawData = occupancy_data_rep , numNei = 5)
nn_roofs = SpatialNeuralNet(rawData = roofs_data_rep , numNei = 5)

nn_years.build_model()
nn_stories.build_model()
nn_foundations.build_model()
nn_occupancy.build_model()
nn_roofs.build_model()

nn_years.train()
nn_stories.train()
nn_foundations.train()
nn_occupancy.train()
nn_roofs.train()

# Surf predicting and saving the files
years_surf = []
stories_surf = []
foundations_surf = []
occupancy_surf = []
roofs_surf = []

for year in years_missing:
    years_surf.append([year[0], year[1],
                      round(nn_years.predict(year))])

for story in stories_missing:
    stories_surf.append([story[0], story[1],
                        round(nn_stories.predict(story))])

```

```

for foundation in foundations_missing:
    foundations_surf.append([foundation[0],
                             foundation[1],
                             nn_foundations.predict(foundation)])

for occupancy in occupancy_missing:
    occupancy_surf.append
    ([occupancy[0], occupancy[1],
     round(nn_occupancy.predict(occupancy))])

for roof in roofs_missing:
    roofs_surf.append([roof[0], roof[1],
                       round(nn_roofs.predict(roof))])

#saving surf results

year_lat = []
year_lon = []
year_pred = []

stories_lat = []
stories_lon = []
stories_pred = []

found_lat = []
found_lon = []
found_pred = []

occ_lat = []
occ_lon = []
occ_pred = []

roof_lat = []

```

```

roof_lon = []
roof_pred = []

for year in years_surf:
    year_lat.append(year[1])
    year_lon.append(year[0])
    year_pred.append(year[2])

for story in stories_surf:
    stories_lat.append(story[1])
    stories_lon.append(story[0])
    stories_pred.append(story[2])

for found in foundations_surf:
    found_lat.append(found[1])
    found_lon.append(found[0])
    found_pred.append(found[2])

for occ in occupancy_surf:
    occ_lat.append(occ[1])
    occ_lon.append(occ[0])
    occ_pred.append(occ[2])

for roof in roofs_surf:
    roof_lat.append(roof[1])
    roof_lon.append(roof[0])
    roof_pred.append(roof[2])

df = list(zip(year_lat, year_lon, year_pred))
Frame=pd.DataFrame(df, columns = ['lat', 'lon', 'surf_year_built'])
Frame.to_csv( "Outputs\Dataset_Results\surf_year_built.csv",
              index=False, encoding='utf-8-sig')

```

```

df = list(zip(stories_lat , stories_lon , stories_pred))
Frame=pd.DataFrame(df, columns = ['lat ', 'lon ', 'surf_num_of_stories '])
Frame.to_csv( "Outputs\Dataset_Results\surf_stories.csv",
              index=False , encoding='utf-8-sig')

df = list(zip(found_lat , found_lon , found_pred))
Frame=pd.DataFrame(df, columns = ['lat ', 'lon ', 'surf_foundation'])
Frame.to_csv( "Outputs\Dataset_Results\surf_foundation.csv",
              index=False , encoding='utf-8-sig')

df = list(zip(occ_lat , occ_lon , occ_pred))
Frame=pd.DataFrame(df, columns = ['lat ', 'lon ', 'surf_occupancy'])
Frame.to_csv( "Outputs\Dataset_Results\surf_occupancy.csv",
              index=False , encoding='utf-8-sig')

df = list(zip(roof_lat , roof_lon , roof_pred))
Frame=pd.DataFrame(df, columns = ['lat ', 'lon ', 'surf_roof_type'])
Frame.to_csv( "Outputs\Dataset_Results\surf_roof.csv",
              index=False , encoding='utf-8-sig')

```

---

Listing B.11: Classifying roof damage state using trained models module

---

```

# NOAA aerial imagery that was downloaded are
# classified into five roof damage states here

#This code was modified
# to run for both trained models

#This is for zoom 18 model
X = []
i=0
while i<=1136:
    img = imread( '''Outputs/Collected_Images

```



```

    /Post_Windstorm_Roofs/{}.png'''.format(i))
    img = img_as_float(img)
    X.append(img)
    i+=1
X = np.array(X)
XS = X.reshape(len(addresses),160000)
X = []

#This is for zoom 20 model
while i<=1562:
    img = imread(''Outputs/Collected_Images
    /Post_Windstorm_Roofs/{}.png'''.format(i))
    img = img_as_float(img)
    X.append(img)
    i+=1
X = np.array(X)
XB = X.reshape(426,160000)

#loading the three
#trained models per zoom level
MLPB = load('Models/Final/MLP_200_by_200_47.27.joblib')
SGDB = load('Models/Final/SGD_200_by_200_40.6.joblib')
SVCB = load('Models/Final/SVC_200_by_200_53.72.joblib')
SVCS = load('Models/Final/SVC_50_by_50_32.94.joblib')

#predicting using all models
MLPB_pred = MLPB.predict(XB)
SGDB_pred = SGDB.predict(XB)
SVCB_pred = SVCB.predict(XB)
SVCS_pred = SVCS.predict(XS)

#Saving the results
df = list(zip(MLPB_pred,SGDB_pred,SVCB_pred))

```

```

Frame=pd.DataFrame(df, columns = [ 'mlpb', 'sgdb', 'svcb' ])
Frame.to_csv( "Outputs\Dataset_Results\BIG_PRED_RES.csv",
              index=False, encoding='utf-8-sig' )

df = list(zip(SVCS_pred))
Frame=pd.DataFrame(df, columns = [ 'svcs' ])
Frame.to_csv( "Outputs\Dataset_Results\SMALL_PRED_RES.csv",
              index=False, encoding='utf-8-sig' )

```

---

### Listing B.12: Saving final dataset results module

---

```

# The final result is the csv file that
# contains the web scraping results
#along with the BRAILS predicted features
# and finally the roof damage prediction
# All are saved

df = list(zip(addresses, lats, lons, stories,
              years, foundations, occupancy,
              roofs, occ_pred, roof_pred, yb_pred,
              no_floors_pred))

Frame=pd.DataFrame(df, columns = ["Address", "Latitude",
                                  "Longitude",
                                  "Stories", "Years",
                                  "Foundations",
                                  "Occupancy",
                                  "Roof_type",
                                  "BRAILS_Predicted_Occupancy",
                                  "BRAILS_Predicted_Roof_Type",
                                  "BRAILS_predicted_year_built",
                                  "BRAILS_predicted_stories" ])

Frame.to_csv
( "Outputs\Dataset_Results\Results_NO-ROOF_DAMAGE_PRED.csv",
  index=False, encoding='utf-8-sig' )

```

---