

USING PASSIVE SEISMIC DATA FROM MULTIPLE  
SENSORS FOR PREDICTING EARTH DAM AND  
LEVEE EROSION

by  
C. Travis Johnson

© Copyright by C. Travis Johnson, 2017

All Rights Reserved

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Computer Science).

Golden, Colorado

Date \_\_\_\_\_

Signed: \_\_\_\_\_

C. Travis Johnson

Signed: \_\_\_\_\_

Dr. Tracy Camp  
Thesis Advisor

Signed: \_\_\_\_\_

Dr. Wendy Fisher  
Thesis Advisor

Golden, Colorado

Date \_\_\_\_\_

Signed: \_\_\_\_\_

Dr. Tracy Camp  
Professor and Head  
Department of Computer Science

## ABSTRACT

Earth dams and levees support industrial applications, flood defense, and irrigation infrastructure around the world. These structures naturally undergo erosion over time and massive erosion events can result in catastrophic failure. When these structures fail, disastrous floods can displace or even kill nearby residents. Thus, it is important to know when and *where* these structures have progressive internal erosion events so appropriate action can reduce the impact of the erosion. This work explores improvements on the performance of previous machine learning methods for the continuous health monitoring of earth dams and levees (EDLs). Specifically, we explore ensemble classification algorithms (Bagging, Boosting, and Random Forest) to combine the passive seismic data from multiple sensors; we note that previous work only considered the data from one sensor in the wired sensor network. By considering features extracted from the signals of multiple sensors, Boosting with support vector machines (SVMs) shows a 1.5 to 41.7% increase in  $F_1$  score over single support vector machine (SVM) models depending on the specific sensor chosen for the single SVM. We also explore the use of SVM models trained on data from distinct sensors for visualizing the locations of detected erosion events in the structure.

## TABLE OF CONTENTS

|   |      |
|---|------|
| ABSTRACT . . . . .  | iii  |
| LIST OF FIGURES . . . . .   | vi   |
| LIST OF TABLES . . . . .  | viii |
| ACKNOWLEDGMENTS . . . . .   | ix   |
| CHAPTER 1 INTRODUCTION AND MOTIVATION . . . . .                     | 1    |
| CHAPTER 2 BACKGROUND . . . . .                                      | 4    |
| 2.1 Machine Learning . . . . .                                      | 4    |
| 2.1.1 Anomaly Detection . . . . .                                   | 7    |
| 2.1.2 Ensemble Methods . . . . .                                    | 8    |
| 2.2 Internal Erosion Events . . . . .                               | 11   |
| 2.3 Related Work . . . . .  | 12   |
| CHAPTER 3 USBR PIPING DATA SET AND INITIAL EXPERIMENT . . . . .     | 16   |
| 3.1 USBR Piping Data Set . . . . .                                  | 17   |
| 3.2 Initial Experiment . . . . .                                    | 20   |
| CHAPTER 4 COMBINING SENSOR DATA USING CLASSICAL ENSEMBLES . . . . . | 24   |
| 4.1 Experimental Setup . . . . .                                    | 25   |
| 4.2 Results and Discussion . . . . .                                | 26   |
| 4.2.1 Bagging Ensembles . . . . .                                   | 26   |
| 4.2.2 Boosting Ensembles . . . . .                                  | 28   |
| 4.2.3 Random Forest Ensembles . . . . .                             | 28   |

|   |  |    |
|---|--|----|
| 4.2.4   | Generalization Abilities of Ensembles . . . . .                      | 30 |
| 4.3   | Analysis of Computational Resource Requirements . . . . .            | 35 |
| CHAPTER 5 COMBINING SENSOR DATA WITH DISTINCT CLASSIFIERS . . . . |  | 38 |
| 5.1   | Distinct Supervised Support Vector Machines . . . . .                | 39 |
| 5.2   | Distinct Semi-Supervised One-Class Support Vector Machines . . . . . | 42 |
| CHAPTER 6 CONCLUSIONS AND FUTURE WORK . . . . .                   |  | 46 |
| 6.1   | Feature Engineering . . . . .  | 47 |
| 6.2   | Number and Placement of Sensors . . . . .                            | 48 |
| 6.3   | Other Data Sets . . . . .  | 48 |
| REFERENCES CITED . . . . .  |  | 50 |

## LIST OF FIGURES

|            |  |    |
|------------|--|----|
| Figure 1.1 | Homes washed away in the aftermath of the failure of the Algodões dam in Cocal da Estação, Brazil. . . . .   | 2  |
| Figure 1.2 | Satellite images of TVA Kingston plant and surrounding area before (left) and after (right) the failure of an earthen retaining wall. . . . .  | 3  |
| Figure 2.1 | The confusion matrix for our anomaly detection problem. . . . .  | 5  |
| Figure 2.2 | Classifications of dam failures. . . . .   | 12 |
| Figure 3.1 | Progression of piping erosion in an EDL. After the pipe initiates, concentrated water flow through the pipe continues the erosion, which can progress to structure failure. . . . .  | 16 |
| Figure 3.2 | Photographs depicting the progression of the piping event observed in the USBR piping experiment. . . . .  | 17 |
| Figure 3.3 | Location of 23 wired geophones installed on the front levee slope in the USBR June 2015 piping experiment (left: photo of the experiment with wired geophone locations circled; right: diagram with dimension labels in feet). . . . . | 18 |
| Figure 4.1 | Diagram showing how one single ensemble trains using the data from $n$ sensors (Option 1). . . . .   | 25 |
| Figure 4.2 | Precision-Recall curves showing the trade off between different probability thresholds for predicting positive class labels. . . . .   | 32 |
| Figure 4.3 | Confusion matrices for all ensembles showing counts of correct and incorrect classifications on the test set by type. . . . .  | 33 |
| Figure 5.1 | Diagram showing how distinct models will train using the data from each sensor (Option 2). . . . .   | 38 |
| Figure 5.2 | Sample visualizations from the predictions of supervised SVM models. Each prediction uses the data from the corresponding geophone sensor. . . . .   | 40 |
| Figure 5.3 | Sample visualizations from the probability predictions of supervised SVM models, where each prediction uses the data from the corresponding geophone sensor. . . . .   | 41 |

Figure 5.4 Sample visualizations from the predictions of semi-supervised OCSVM models, where each prediction uses the data from a distinct geophone sensor. . . . . 44

## LIST OF TABLES

|           |   |    |
|-----------|---|----|
| Table 2.1 | Possible performance measures for classification problems. . . . .  | 6  |
| Table 3.1 | Features extracted from windowed passive seismic data. . . . .  | 19 |
| Table 3.2 | Performance metrics for each support vector machine trained using a different sensor's passive seismic data. . . . .  | 23 |
| Table 4.1 | Hyper-parameter values selected through grid search cross-validation using <code>GridSearchCV</code> class for the <code>BaggingClassifier</code> and $F_1$ score as the cross-validation metric. . . . .                                 | 27 |
| Table 4.2 | Hyper-parameter values selected through grid search cross-validation using the <code>GridSearchCV</code> class with the <code>AdaBoostClassifier</code> as the base classifier and $F_1$ score as the cross-validation metric. . . . .    | 28 |
| Table 4.3 | The top twenty features used for splitting observations in decision trees of the <code>RandomForestClassifier</code> . . . . .  | 30 |
| Table 4.4 | Performance metrics for each ensemble trained using passive seismic data from all 23 sensors in the USBR piping experiment. . . . .   | 31 |
| Table 4.5 | The average time and memory to fit training set (of 540 observations) and to predict class labels for the test set (of 180 observations) with 100 repetitions of each ensemble in Python (times in seconds; memory in Megabytes). . . . . | 36 |

## ACKNOWLEDGMENTS

I would like to thank Dr. Tracy Camp for the countless learning opportunities she provided throughout my time at Mines: whenever I re-established my comfort zone, she pushed me right back out to grow again. This project would not have succeeded without the work and guidance of Dr. Wendy Fisher. I would also like to thank Zoe Nacol and our dog Chuck for their emotional support as I completed my M.S. degree. Lastly, I thank my incredible parents, Pam and Carey Johnson, for always providing me with the means to achieve my dreams.

# CHAPTER 1

## INTRODUCTION AND MOTIVATION

Our study explores machine learning techniques for the continuous monitoring of earthen structures using passive seismic data. Earth dams and levees (EDLs) are structures built of clay, dirt, sand, and rock designed primarily to control flooding and provide water for irrigation. Since these earthen structures, by design, retain water, they naturally undergo weathering and erosion processes. Erosion inside the structure is internal erosion and comes in different forms with varying severity (see Chapter 2.2 for a detailed discussion). We utilize passive seismic data collected from *multiple* sensors installed along earthen structures to detect these internal erosion events.

Previous work has demonstrated the value of employing machine learning systems for monitoring EDLs; however, these studies have only utilized the passive seismic data from *one* sensor installed in the structure rather than all the data available from the grid of sensors [1–3]. Machine learning methods are typically most successful in data rich environments [4], so we intuitively expect that providing more data to our machine learning models will yield more reliable, consistent, and useful results.

A real world example that motivates our study to design health monitoring systems for EDLs is the bursting of the Algodões Dam in northeast Brazil. This dam failed just weeks after the state engineer assured the public the dam was safe. A 50 meter (164 feet) wide hole opened in the dam, and the resulting flood, depicted in Figure 1.1, submerged the city of Cocal da Estação under approximately 20 meters (66 feet) of water. This catastrophe displaced 30,000 residents, destroyed homes, and drowned at least six people [5].

Continuous monitoring systems, like those discussed in our study, augment the current method of monitoring EDL structures, i.e., where an expert does periodic visual inspections. With human experts, there is no guarantee that the inspection will detect internal erosion



Figure 1.1: Homes washed away in the aftermath of the failure of the Algodões dam in Cocal da Estação, Brazil. Figure from [5].

soon enough to take appropriate action. Thus, a continuous monitoring system is necessary to detect internal erosion as soon as possible, allowing time to take corrective action and possibly execute evacuation plans.

A second real world example of an earthen structure failing, after an expert inspection deemed the structure healthy, is the disastrous coal fly ash slurry spill at the Tennessee Valley Authority (TVA) Kingston Fossil Plant in 2008. The failure of an earthen retaining wall released approximately 4.1 million cubic meters (about 1 billion gallons) of coal fly ash slurry which covered homes and the surrounding area as shown in Figure 1.2. Environmental studies completed in the aftermath of the spill found that high levels of mercury and other metals in the nearby Emory and Clinch Rivers posed long-term threats for the ecological systems [6]. The day before failure, a regular inspection found no signs of erosion although the root cause analysis attributed the failure, in part, to seepage erosion [7].

These examples illustrate that failures of earthen retaining structures can have disastrous consequences, and expert visual inspections do not always catch potential problems. Automatic monitoring systems strive to mitigate the results of disasters by providing early warnings of potential problems. We study approaches to implementing such monitoring systems. Our specific research question is whether utilizing data from multiple geophone



Figure 1.2: Satellite images of TVA Kingston plant and surrounding area before (left) and after (right) the failure of an earthen retaining wall. The rectangle overlaid on each image outlines the retaining wall which failed. Figure adapted from [8].

sensors will improve performance of anomaly detection systems.

## CHAPTER 2

### BACKGROUND

We begin with an overview of machine learning and anomaly detection. We then present the ensemble methods in machine learning which we use in Chapter 4 to combine data from multiple sensors for our particular problem. This chapter concludes with a discussion on the different types of internal erosion events that exist, as well as related work that has studied detecting erosion events in EDLs.

#### 2.1 Machine Learning

Machine learning methods aim to improve the performance of computing systems automatically through experience, where experience is most typically *data* [9]. As outlined by Tom Mitchell, the components of a well-defined machine learning problem include the task, performance measure, and experience for the learner. Our study aims to apply learning methods to the following machine learning problem:

**Task,  $T$**  Decide whether the state of an earthen structure, like an EDL, is normal for a particular period of time. If possible, identify the location of the abnormal state in the structure.

**Performance Measure,  $P$**  Use  $F_1$  scores to compare performance of learners as is common in classification problems. Figure 2.1 presents the confusion matrix and the acronyms used in Table 2.1 to define all performance metrics discussed in our study. In practice, it may be wise to use an  $F_\beta$  score to account for self-healing events and the larger cost of erosion events going undetected compared to false alarms. In other words, if we use  $F_\beta$  instead of  $F_1$  for performance evaluation, then  $\beta$  needs to favor recall over precision (e.g.,  $\beta > 1$ ).

**Experience,  $E$**  The experience for our learners is passive seismic data collected from a network of geophones. Geophones are sensors that measure ground displacement using voltage readings. These voltage readings are proportional to the velocity of elastic waves propagating through the subsurface of the earth [10]. Typically, we have more reliable labels and more observations of normal events than anomalies. In other words, we have more examples of dams in safe operation than examples of dams failing or dams with confirmed internal erosion events.

**Confusion Matrix**

|                   |          | <b>Predicted Label</b>                                       |  |
|-------------------|----------|--|--|
|                   |          | Positive   | Negative   |
| <b>True Label</b> | Positive | True Positive<br>(TP)<br>Detected Erosion<br>Event Correctly | False Negative<br>(FN)<br>Erosion Event<br>Missed                |
|                   | Negative | False Positive<br>(FP)<br>False Alarm<br>No Erosion<br>Event | True Negative<br>(TN)<br>Detected Safe<br>Operation<br>Correctly |

Figure 2.1: The confusion matrix for our anomaly detection problem. A positive label (1) denotes an EDL undergoing internal erosion events while a negative label (0) represents an EDL in safe operation. Ground truth data determines the true labels for observations; we note that ground truth data is not always available. Predicted labels are the outputs from our machine learning models. Correct predictions are those where true and predicted labels match.

Machine learning methods, or models, improve through experience by “learning” parameters of functions  $f(x)$  to map training examples, or feature vectors,  $x$  to some target output  $y$ . We learn  $f(x)$  by extracting patterns from a large matrix,  $X$ , of training examples. Then, we evaluate the “fitness” of the learned hypothesis function by comparing the predictions for test examples with true labels (as shown in Figure 2.1). The  $y$  target can be either a

Table 2.1: Possible performance measures for classification problems. Each metric, restricted on the range  $[0, 1]$ , becomes more optimal as its value approaches 1. The confusion matrix in Figure 2.1 defines the TP, TN, FP, and FN acronyms.

| Performance Measure | Description   |
|---------------------|---|
| Accuracy            | Percentage of correctly classified observations; accuracy is not a good performance metric for imbalanced data sets, because learning to always predict one label could improperly result in high accuracy. $\frac{TP+TN}{\text{All Predictions}}$                            |
| Precision           | Measures the correctness of our positive predictions [11]. In other words, how often there actually was an anomalous event when we predicted an anomalous event. $\frac{TP}{TP+FP}$   |
| Recall              | Measures the fraction of positive observations predicted as positive [11]. In other words, how many of the total anomalous events did we successfully detect. $\frac{TP}{TP+FN}$  |
| $F_1$ score         | The harmonic mean of precision and recall. $F_1$ score combines the metrics of precision and recall into a single metric.   |
| $F_\beta$ score     | The weighted mean of precision and recall, which is similar to $F_1$ score except the $\beta$ parameter allows for weighting precision and recall in the combined $F$ score. As $\beta$ increases, $F_\beta$ favors recall. As $\beta$ decreases, $F_\beta$ favors precision. |

continuous or discrete response variable. In the continuous case, we call the machine learning methods “regression models”; on the other hand, methods that learn discrete responses are called “classification models.” In our study, we apply classification models to predict discrete class labels (normal or anomaly) and draw knowledge from the subfield of anomaly detection (discussed in Chapter 2.1.1).

We use machine learning methods in our study to extract the normal patterns from experience  $E$  so we can perform task  $T$ . In other words, we seek to learn general functions that can map passive seismic data readings to labels of *normal* or *anomaly*. By doing so, we “teach” continuous monitoring systems what to consider normal. Then, as the systems detect anomalous observations, we can raise alarms for expert analysts. An automatic detection system can assist human experts tasked with conducting regular inspections of the EDLs by directing their focus to the most urgent problems.

### 2.1.1 Anomaly Detection

Anomaly detection is a commonly studied application of machine learning where the goal is to learn what characterizes normal events in a system so we can discern what is *not normal* behavior for the system. Our particular application of anomaly detection is for identifying abnormal events in passive seismic data from a grid of geophones on the surface of an EDL. Other, arguably more common, applications of anomaly detection include fraud detection, in both insurance and finance realms, as well as intrusion detection in cybersecurity [12]. A variety of strategies exist for anomaly detection, and each makes its own unique assumptions about the underlying problem and data set. Chandola et. al. describe these strategies and assumptions in detail [12]. For reference, the assumptions made by some of the methods used in related EDL anomaly detection work are included next.

**Classification-Based** The normal and anomalous classes are separable in the given feature space. In machine learning, a *feature space* is simply the mathematical space defined by the measured characteristics of observations. Our particular feature space is described in Chapter 3.1.

**Nearest Neighbor-Based** Dense neighborhoods of observations form the normal classes while anomalies are the observations far from their closest neighbors.

**Clustering-Based** Normal observations occur in clusters while anomalies do not belong to any cluster. The distinction between nearest neighbor and clustering methods is that nearest neighbor compares an observation and its closest neighboring observations whereas clustering compares an observation and its closest cluster center.

**Statistical-Based** In stochastic models, normal observations are those that are more probable to occur in the models.

Supervised, unsupervised, and semi-supervised algorithms exist for anomaly detection, similar to the algorithms available for traditional classification problems (e.g., spam detec-

tion). The type of algorithm to use depends on the number of ground truth labels available for the data set. Anomaly detection problems typically differ from other classification problems in one or both of the following ways:

- Examples of anomalies are more rare than examples of the normal class, because anomalies are inherently less frequent. Thus, data sets for anomaly detection problems are highly imbalanced.
- Obtaining accurate labels, especially for the anomalous class, can be challenging or prohibitively expensive. Thus, data sets may not represent all cases of abnormal observations.

Anomaly detection techniques lend themselves to our research problem since our data sets exhibit a class imbalance and we do not have examples of all possible erosion events in EDLs [3]. We experiment with supervised anomaly detection using the data from a laboratory earth embankment described in Chapter 3.1. We first investigate a supervised approach to evaluate the performance of the models developed. Specifically, we experiment with classical ensemble methods (presented in Chapter 2.1.2) for supervised anomaly detection. Additionally, we investigate the development of an anomaly detection strategy that combines multiple supervised or semi-supervised models to provide visualizations on the location of an internal erosion event in an EDL.

### **2.1.2 Ensemble Methods**

In machine learning, ensembles seek to improve the overall performance of predictive models by combining the predictions of multiple base classifiers (e.g., artificial neural networks, logistic regression, decision trees, support vector machines). To understand the intuition behind ensemble methods, consider a college admissions committee faced with accepting or rejecting applications. Each member of the committee is making a decision about acceptance based on the same information, but the decision process for each committee member (e.g., base classifier) differs slightly. In general, we expect the committee to make better decisions

as a group than as individuals. In the same way, machine learning ensembles bring together slightly different models to collaborate for an overall better result.

In short, we build ensembles in two stages. First, we train diverse learning models individually, then we combine their predictions [13]. By doing this, it becomes possible to achieve better performance on our learning task; however, there is always a danger of creating models that are too complex, or *overfit*<sup>1</sup> the data. Ensemble methods create more complex models in order to reduce bias, variance, or both and usually provide better generalization to out-of-sample data [13].

The classic ensemble techniques in machine learning literature are Bagging, Boosting, and Random Forest [13]. All three types of ensembles belong to a class called *importance sampling learning ensembles*, which is a class first outlined by Friedman and Popescu [15]. The idea is that we can sample the training sets in ways that provide more useful information to our learners. Then, these diverse learners cooperate to generalize well on the original problem [15].

Bagging (**bootstrap aggregating**) is an importance sampling ensemble with a simple sampling technique. Bagging uses bootstrap replicates of the original training set for training each of the base classifiers in the ensemble, where a bootstrap replicate is simply a random sample drawn with replacement. Bagging is typically most useful with *unstable* learning algorithms that learn dramatically different hypotheses when trained with minor changes to the training set. Bagging reduces variance problems in base classifiers with little impact on bias [13, 16].

Boosting, specifically Adaptive Boosting (AdaBoost), uses a different sampling strategy for enforcing diversity into the training sets for each base classifier. In an iterative training process, weights associated with each of the observations in the training set update according

---

<sup>1</sup>Overfitting occurs when a model learns the noise or unique characteristics of the training set so closely that it does not generalize. In other words, overfit models perform well on training sets but not on new observations. The converse of an overfit model is an underfit model which does not perform well on the training set [14]. To balance the trade-off of overfitting and underfitting, machine learning, in practice, adopts the philosophy of Occam's razor that promotes the virtue of simplicity [4]. A simple hypothesis that performs comparably to a more complex hypothesis is preferable.

to the current level of error the particular observation is causing in the ensemble. A higher weight implies that the example is more likely sampled for the next training set, which helps ensure each additional base classifier focuses on correctly classifying the examples that the rest of the ensemble predicts most erroneously [17]. The iterative training process continues until we either reach a desired error level or exceed some predefined limit of base classifiers in the ensemble. Boosting is typically used with weak classifiers (e.g., those that perform barely better than random) and might not be the best choice for use in noisy data sets [17]. That is, in a noisy data set, Boosting will focus on fitting the outliers rather than refining the good hypotheses with the quality data.

Another importance sampling ensemble method is Random Forest, which does sampling of the *features* of a training set rather than sampling of the *observations* [18]. Random forest implementations often provide options for using bootstrap replicates, similar to Bagging, when training decision trees for the Random Forest. These options allow for sampling of both observations and features, which is possible with the `RandomForestClassifier` in `scikit-learn` [19]. One other difference with Random Forests, when compared to Bagging and Boosting, is the fact that Random Forests are not meta-learners. That is, Random Forests always use decision trees as their base classifiers. Random Forests often exhibit comparable performance to AdaBoost and other Boosting methods; however, Random Forests are typically more robust to noisy data sets [18].

Some ensemble techniques focus directly on inducing diversity in the base classifier decisions. For instance, the DECORATE ensemble technique induces diversity by adding artificial data to training sets with labels that oppose the current ensemble’s predictions [20], as diversity is fundamental to the success of ensemble techniques. Imagine an ensemble with no diversity in the hypotheses learned by the base classifiers composing the ensemble. In this case, we have not improved our performance but have incurred the overhead of training, storing, and asking for new predictions from redundant copies of the model. This contrived example demonstrates the importance of diversity among base classifiers in

ensemble methods.

Chapter 3 presents initial results that show models trained on the data from different sensors in a laboratory experiment learn diverse hypotheses. We investigate the classic importance sampling ensemble methods (Bagging, Boosting, and Random Forest) with the expectation that they will improve performance of current continuous monitoring systems. We also investigate an approach for using passive seismic data from multiple sensors to locate erosion events in EDLs by training distinct models for each sensor.

## 2.2 Internal Erosion Events

Figure 2.2 shows there are a variety of different erosion/failure events that can impact the health of an EDL. According to a 2011 study of dam failures in the United States, the most common failure types are overtopping (B), foundation seepage (typically caused by E, I, or J), and piping (G). The study estimated that 34% of dam failures were due to overtopping, 30% due to foundation seepage, 20% due to piping, and 10% due to “problems with conduits and valves” [21]. The study attributed a small percentage of the failures to disasters such as earthquakes, extreme storms, and sabotage. The continuous monitoring systems resulting from our study are not capable of providing timely warnings for all types of failures. However, the majority of dam failures are due to internal erosion events that could be detected before total dam failure, perhaps providing warning of dam failure in at least 50-70% of cases.

An erosion event in an EDL can take on different forms and it is difficult and costly to measure real instances of each different failure type. Therefore, detecting erosion events is an exemplar anomaly detection problem. Regardless of the form of erosion inducing abnormal examples in the data set, erosion induced anomalies are of interest to analysts [12]. Some anomalies can be noise, which are abnormal examples that are not of interest to the analyst studying the EDL. Sources of noise in our data sets could be cars and large trucks on nearby roads or perhaps birds landing on the tops of sensors. Noise removal techniques often improve results when used prior to anomaly detection [12].

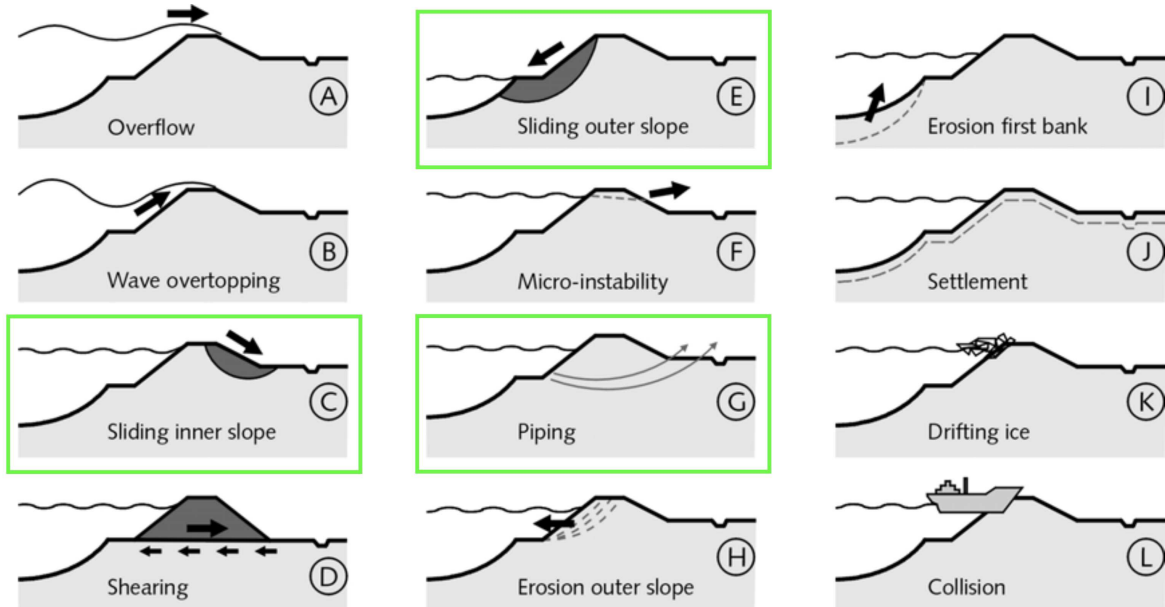


Figure 2.2: Classifications of dam failures, figure reproduced from [22]. We have outlined the three forms of erosion observed in the laboratory experiment described in Chapter 3.1. The experiment observed significant piping (G) and sliding of the inner slope (C) with some sliding of the outer slope (E).

An interesting complication to the problem of anomaly detection of internal erosion events in EDLs follows: sometimes the internal erosion events undergo “self-healing.” For instance, a pipe event heals itself through a collapse or clogging of the pipe [23]. This implies that false positives will occur naturally. That is, our detection method may raise an alarm about an internal erosion event in a structure only to have the erosion heal before a more detailed inspection occurs. We prefer to raise false alarms than fail to detect erosion events that develop into larger problems. In short, we prefer to have more false positives than false negatives.

### 2.3 Related Work

Previous studies have shown the usefulness of anomaly detection methods for monitoring EDLs. Our study builds upon previous work at the Colorado School of Mines that applied machine learning techniques to the data from a single sensor in experimental data sets. First,

researchers applied clustering techniques for an unsupervised learning approach to anomaly detection in EDLs using passive seismic data [1]. Four of the five clustering techniques used in the study demonstrated a separation of the two event types (normal and anomalous). The best performing clustering technique, Hierarchical Clustering, identified clusters with a maximum purity of 83.8%. That is, approximately 83.8% of the data points were associated with the correct cluster.

Follow-up studies using passive seismic data trained support vector machines (SVMs) in both supervised and semi-supervised modes [2, 3]. The supervised SVM approach treats the problem as a two-class classification: the SVM trains with data labeled for both categories (i.e., “normal” or “anomalous”). The semi-supervised SVM approach treats the problem as a one-class anomaly detection problem [24]: the SVM trains only with data labeled as “normal” and seeks to find some surface that bounds the group of normal observations. Any new observation not inside the surface is then deemed an anomalous event [3]. Again, the results demonstrated a separation of the two event types, though room exists for improving the performance metrics of the learning algorithms. The two-class SVM resulted in an accuracy score of approximately 97% while the one-class SVM was only able to achieve approximately 83% accuracy. After adding Haar wavelet transformations to reduce noise, the one-class SVM’s accuracy improved to 91% [2]. The study concludes that, while the two-class SVM’s performance is better than the one-class SVM on the experimental data set, the one-class SVM is more suitable for the real world problem of anomaly detection in EDLs due to the large imbalance of our data sets.

Studies at the Colorado School of Mines have also explored the use of a more statistical approach, using a multivariate Gaussian strategy for anomaly detection in passive seismic data from EDLs [25]. The multivariate Gaussian (MVG) approach fits a Gaussian distribution to a training set assumed to solely consist of normal data.<sup>2</sup> Then, we compute a probability that a new observation belongs to the fitted Gaussian distribution. By threshold-

---

<sup>2</sup>It is possible for anomalous observations to leak into the training set for an MVG. For example, a ground truth label could be incorrect.

ing the probability, the MVG predicts a class label as normal or anomalous. This approach achieved 97% accuracy in correctly labeling experimental observations. This study, however, only considered the data from a single sensor deemed best for experimentation despite there being more data to use. In the real application of a continuous monitoring system, we won't know which one sensor is best to rely on for detecting anomalies, e.g. we won't know which sensor is closest to the anomaly. Our study attempts to ameliorate this problem.

Other institutions have also studied anomaly detection for structural health monitoring of EDLs. At the University of Amsterdam, researchers installed sensors in earth levees to measure properties of the levee in real time as part of the UrbanFlood project seeking to bring scalable, continuous monitoring systems to Europe's flood defense infrastructure [26]. Properties measured included pore water pressure, temperature, and inclination. With a one-sided classification approach, the researchers were able to detect anomalous events in their experimental levees; however, their approach requires installing sensor systems inside the EDL structure [27]. Our sensor systems (a grid of geophones placed on top of the existing structure) retain the integrity of the EDL. Furthermore, our sensor networks consist only of geophones whereas their studies relied on different types of sensors used to train disparate learning models.

Researchers at Mississippi State University employed a support vector machine model to detect anomalous events (e.g., "levee slides") along the Mississippi river with remote sensing techniques. Specifically, they used data from the NASA JPL Uninhabited Aerial Vehicle Synthetic Aperture Radar (UAVSAR) and were able to learn hypotheses with accuracies ranging from 93-95% [28]. The benefit of using remote sensing techniques is that they are non-intrusive; however, sensors installed on (or in) the physical structures typically provide more accurate and reliable measurements [27]. Furthermore, a network of geophones is cheaper for a continuous monitoring system since the UAVSAR costs upwards of \$2,500 per flight hour and requires submission of flight plans to JPL in advance of the flight [29]. As a comparison, the sensors used in our experiment cost around \$150 each [30]. The geophones

also monitor continuously with a much lower cost per hour. The remote sensing techniques provided visualizations to locate the sliding slopes of levees (Figure 2.2 shows sliding erosion events). We seek to accomplish a similar goal with more readily available and cheaper geophone sensors.

### CHAPTER 3

#### USBR PIPING DATA SET AND INITIAL EXPERIMENT

This chapter describes the data set studied along with an initial experiment to demonstrate the diversity of decision functions learned using data from different sensors installed in a laboratory EDL. The main data set considered in this work comes from the United States Bureau of Reclamation, where an experiment in 2015 recorded passive seismic data from a small scale earthen levee. In the experiment, researchers removed a piece of rebar built into the structure after the head of the reservoir was steady. This experiment sought to better understand piping events (shown in Figure 3.1) as they were responsible for approximately 20% of EDL failures in 2011 in the United States [21]. In addition to inducing a direct piping event, the study recorded seepage and crack events that occurred after the piping began; Figure 3.2 depicts the progression of the piping event observed which reflects the understanding of piping shown in Figure 3.1.

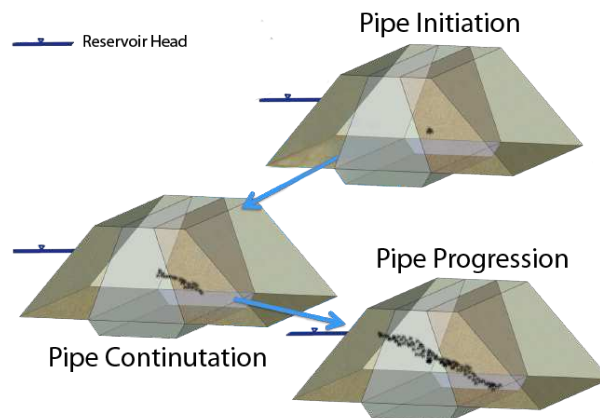


Figure 3.1: Progression of piping erosion in an EDL. After the pipe initiates, concentrated water flow through the pipe continues the erosion, which can progress to structure failure. Figure adapted from [31].

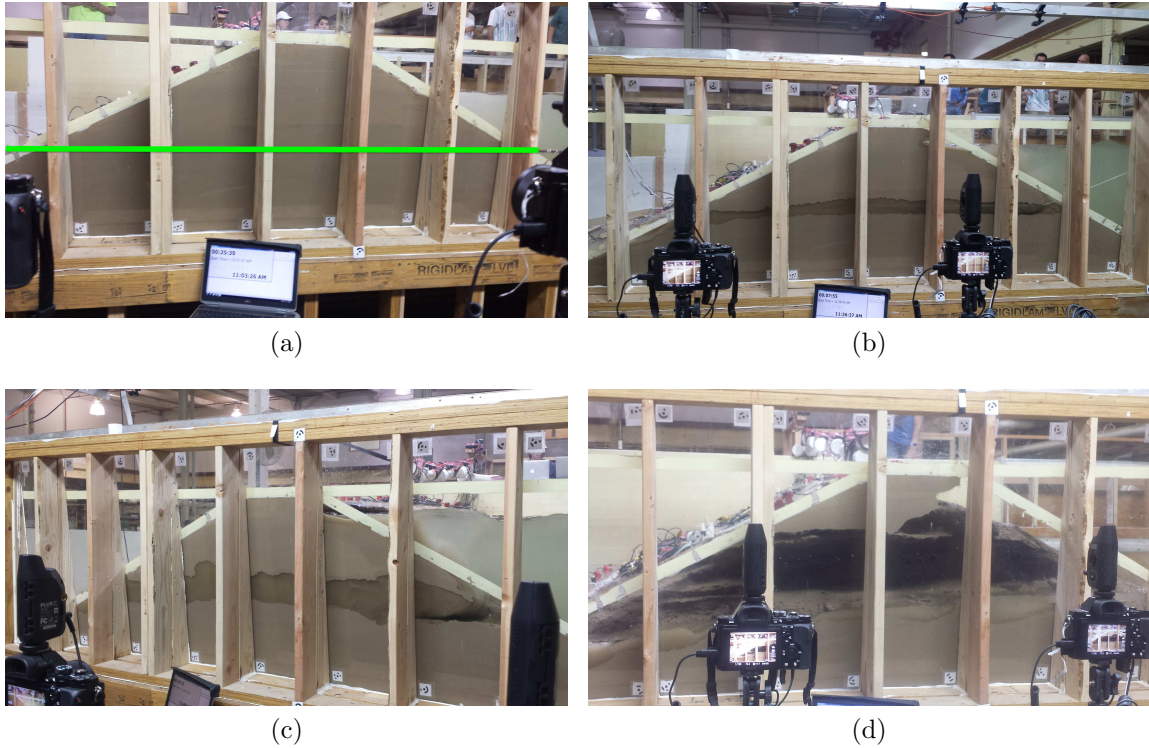


Figure 3.2: Photographs depicting the progression of the piping event observed in the USBR piping experiment. (a) shows the structure before the experiment began with the overlaid line denoting the location of the rebar; (b) shows the structure soon after removing the rebar to induce the piping event; (c) shows the pipe growing as water flows through the structure; (d) shows the end of the experiment. Despite the massive erosion visible by the end of the experiment, the structure was not brought to complete failure during the experiment because the downstream reservoir filled.

### 3.1 USBR Piping Data Set

We chose this data set since we can identify the location of the erosion events in the structure. Thus, this data set can validate spatial visualizations of the anomalies detected. Other available passive seismic EDL data sets have less quality field notes on the location of erosion events. The future work section, Chapter 6, describes these other data sets. Figure 3.3 presents the locations of each sensor in the experiment; the figure also shows that the rebar used to induce the piping event was closest to sensors 1, 11, 12, and 23.

The passive seismic data collected from this experiment is of a continuous, time-series nature. Machine learning models, however, typically require discrete observations of some

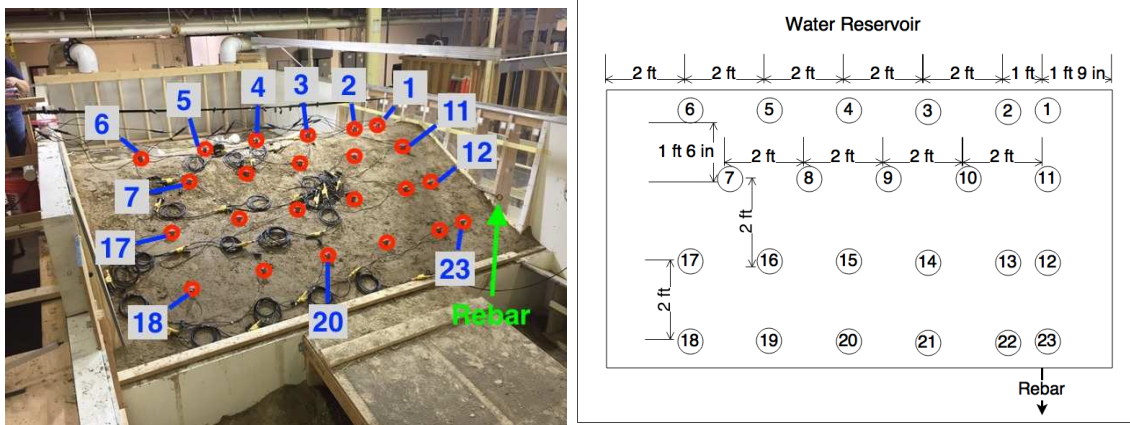


Figure 3.3: Location of 23 wired geophones installed on the front levee slope in the USBR June 2015 piping experiment (left: photo of the experiment with wired geophone locations circled; right: diagram with dimension labels in feet). A piece of rebar was originally built into the structure below wired geophone number 11 and then removed to induce a piping event followed by other erosion events (e.g., sliding and seeping).

feature space. Therefore, we discretize our passive seismic data by binning the data into 3 second time windows, or frames, and extract features from each of those 3 second frames using the open source `MIRToolbox` in `MATLAB` [32]. We leverage previous work and use 3 second frame lengths throughout our study [3]. Fisher et al. found that frame sizes of 1, 2, 3, 5, and 10 seconds all yielded similar results when used for training support vector machine classifiers [3]. The work of M. Rubin influenced the features chosen for our study; he determined that features common in audio signal processing suffice as a summary of the spectral frames [30]. An example of a spectral feature extracted is `zerocross`, which measures the number of times the signal changes sign in the 3 second frame. Table 3.1 lists all features extracted from each frame of the raw passive seismic data.

Similar to Fisher et. al. [2, 25], we apply level three Haar wavelet transforms to the raw passive seismic data prior to frame decomposition and feature extraction. Haar wavelet transforms reduce noise in our data set by decomposing the signal into discrete wavelet coefficients, eliminating those coefficients that fall below a certain level, and then applying the inverse transform to return our signal to its original domain. Thus, Haar wavelets can

improve our system’s ability to discern anomalous data (erosion events) from noisy data [2, 25].

Table 3.1: Features extracted from windowed passive seismic data, with descriptions adapted from [3, 32, 33].

| Feature                 | Description   |
|-------------------------|---|
| Zerocross               | The rate at which the sign of the signal changes, or crosses the zero axis. Provides a temporal indication of the noisiness of the signal.  |
| Spread                  | The standard deviation of the signal, also called the second central moment, which provides a statistical description of “brightness” in audio signals.   |
| RMS Energy              | Measures the energy of the signal; the root mean square of the amplitude.   |
| Rolloff                 | The frequency limit which contains 85% of the total signal: estimates the amount of high frequency energy in the signal.  |
| Flatness                | An indication of the smoothness (or spiky-ness) of the distribution; a ratio of geometric and arithmetic means.   |
| Kurtosis                | Called the fourth standardized moment; a statistical descriptor of the signal distribution that can indicate peaks in the data.   |
| Irregularity            | Measures the degree of variation of successive peaks in the spectrum of a signal.   |
| Coefficient of Skewness | Skewness is the third central moment in signal processing and measures the symmetry of the signal distribution. The coefficient of skewness is simply skewness divided by standard deviation cubed. A symmetrical distribution will have a coefficient of skewness of zero. |

Class labels for both the normal and anomalous data observations originate from the lab notes, pictures, and other data collected during the experiment: geophysicists that attended the USBR piping experiment verified the ground truth labels. The anomalous events labeled include sloughing (sliding of both outer and inner slope) and foundation seepage after the piping event in the test structure began. These labels allow us to train supervised learning models on the observations extracted from the original signal.

For our initial experiment, we used the same set of training and test labels for each of the  $n$  classifiers. There may be an advantage to uniquely crafting a set of observation labels for each of the sensors discussed in Chapter 5. We ask - is it logical to label data from sensors far from the erosion event as anomalous? On one hand, there is an anomaly occurring in

the structure that we want to detect. On the other hand, if we want to spatially locate an anomaly, then our system likely requires more carefully crafted label sets for each sensor.

### 3.2 Initial Experiment

As discussed in Chapter 2.1.2, diverse predictions of base classifiers in ensembles is crucial. We trained classifiers using data from all the geophones in the sensor network and calculated the diversity in the predictions of the classifiers to ensure it is possible to improve the performance of anomaly detection in passive seismic data using ensemble techniques.

We adopt and present the diversity measure used by Melville and Mooney in their paper on the DECORATE ensemble algorithm [20]. Let  $C_i(x)$  denote the  $i$ th classifier in the ensemble’s prediction for observation  $x$  (0 meaning normal and 1 meaning anomalous) and let  $C^*(x)$  be the entire ensemble’s prediction for observation  $x$ . We then measure the disagreement of a classifier with the entire ensemble,  $d_i(x)$ , as follows:

$$d_i(x) = \begin{cases} 0, & \text{if } C_i(x) = C^*(x) \\ 1, & \text{otherwise} \end{cases} \quad (3.1)$$

Using this equation, we compute the diversity metric as the probability that any of the  $n$  classifiers disagree with the predictions of the entire group for all  $m$  observations in the test set:

$$\text{Diversity} = \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m d_i(x_j) \quad (3.2)$$

For our initial experiment, we train one classifier (support vector machine) on the passive seismic data from each of the  $n$  sensors for a total of  $n$  classifiers. Then, we define the ensemble prediction  $C^*(x)$  for any observation  $x$  as

$$C^*(x) = \begin{cases} 0, & \text{if } \sum_{i=1}^n C_i(x) < \frac{n}{2} \\ 1, & \text{otherwise} \end{cases} \quad (3.3)$$

That is, we take the majority label (either 0 or 1) predicted by all  $n$  classifiers to be the prediction of the ensemble.

In our experiment, we extracted features for 3 second frames from 2160 seconds (36 minutes) of the experiment after the piping event began. Of the resulting 720 observations per sensor, 540 (75%) comprised the data set used for training and cross-validation<sup>3</sup> and 180 (25%) comprised the test set for evaluating our method’s generalization ability. The training set consisted of approximately 444 (82%) normal observations and 96 (18%) anomalous observations. The test set consisted of approximately 143 (79%) normal observations and 37 (21%) anomalous observations. We hypothesize that models trained with data from different sensors would learn diverse decision functions demonstrating that the different sensors present different information for classifiers to learn.

In addition to the frame decomposition and feature extraction, we scale feature values and select a subset of features using Python’s `scikit-learn` framework [19]. Feature scaling, using the `StandardScaler` class, ensures that each feature in the data set has a zero mean and unit standard deviation to enforce the constraint that all features have the same magnitude. This scaling can be beneficial to some learning algorithms, especially linear models like kernelized support vector machines [14]. The main reasons SVMs benefit from feature scaling are 1) features in larger ranges will not dominate those in smaller ranges and 2) inner products computed by kernel functions can avoid numerical problems [34]. After feature scaling, we used the `SelectFromModel` class to determine an optimal subset of the features for training each support vector machine model.

Lastly, we utilized 5-fold, stratified cross-validation<sup>4</sup> to select the best kernel for each model using the `GridSearchCV` class in `scikit-learn`. The models trained on the data from sensors 18 and 19 found the radial basis function (RBF) provided better performance

---

<sup>3</sup>Cross-validation, used within our training process, can search for the optimal values for *hyper-parameters* of the machine learning models without leaking test set information into the training process [14]. Specifically, we break the training set into  $k$  folds of approximately equal size. Then, we train  $k$  models on each permutation of  $k - 1$  folds for training and 1 fold for evaluation. We average the evaluation of each model to report an overall cross-validation score and choose the hyper-parameters that maximize the cross-validation score. Hyper-parameters are the parameters for machine learning models that are manually set rather than automatically learned from the data. Two example hyper-parameters include the type of kernel to use for a support vector machine and the maximum depth of a decision tree.

<sup>4</sup>Stratified cross-validation is an extension of  $k$ -fold cross-validation that ensures the folds of the training and validation sets have similar class distributions [14].

than a linear kernel through cross-validation. For the rest of the SVMs trained, the linear kernel proved to be the optimal kernel based on cross-validation results. However, when we move to the one sided (semi-supervised) classification approach, a RBF kernel is commonly used [19] and, with the correct kernel parameters, can find a separation between classes in any data set [24]. Table 3.2 shows the performance metrics evaluated on the test data set for each model trained on a distinct sensor’s data. The table includes the  $F_\beta$  scores for each model with  $\beta = 2$ . We chose to show results for  $\beta = 2$ , as  $\beta = 2$  places more weight on recall than precision. Note the  $F_\beta$  scores reported in Table 3.2 are consistently lower than the  $F_1$  scores. This result shows why a dam operator may investigate  $F_\beta$  scores with  $\beta > 0$ , to emphasize the importance of minimizing false negatives (which are more expensive errors than false positives).

We then calculated the diversity metric (see Equation 3.2) for the collection of classifiers based on their predictions for the test data set. The diversity metric evaluates to approximately 0.32, which means there is a *32% chance that any single classifier disagrees with the majority vote of all classifiers for any given test observation.*

This initial experiment shows that there is diversity among the decision functions (hypotheses) learned by models trained using the data from sensors located at different distances from the anomalies. Thus, we may find better results in our anomaly detection system by employing ensemble techniques to exploit this diversity. Furthermore, these results suggest that we cannot confidently rely on the predictions from a learner that is only using data from a single sensor in the structure.

Table 3.2: Performance metrics for each support vector machine trained using a different sensor’s passive seismic data. The best result in each column is in bold. The worst value in each column is underlined. Sensor 11 was one of the closest sensors to the erosion events in the experiment, which helps explain why model 11 demonstrated best performance in 4 of the 5 reported metrics. The metrics reported for model 11 in our experiment are all within 5% of the metrics for the similar two class SVM research reported by Fisher et. al. [2]. The models trained for sensor numbers 15 and 20 exhibited the worst performance when, by intuition, we expected models trained on data farthest from the rebar (numbers 6, 17, and 18) to perform the worst. We plan to investigate this surprising result further in future work.

| Model Trained on<br>Data from<br>Sensor Number | Accuracy     | Precision    | Recall       | $F_1$        | $F_\beta (\beta = 2)$ |
|--|--------------|--------------|--------------|--------------|-----------------------|
| 1  | 0.950        | 0.912        | 0.838        | 0.873        | 0.852                 |
| 2  | 0.950        | 0.938        | 0.811        | 0.870        | 0.833                 |
| 3  | 0.961        | 0.941        | 0.865        | 0.901        | 0.879                 |
| 4  | <b>0.972</b> | <b>1.000</b> | 0.865        | 0.928        | 0.889                 |
| 5  | 0.956        | 0.968        | 0.811        | 0.882        | 0.838                 |
| 6  | 0.928        | 0.900        | 0.730        | 0.806        | 0.758                 |
| 7  | 0.917        | 0.893        | 0.676        | 0.769        | 0.710                 |
| 8  | 0.944        | 0.966        | 0.757        | 0.848        | 0.791                 |
| 9  | 0.944        | 0.935        | 0.784        | 0.853        | 0.810                 |
| 10   | 0.944        | 0.909        | 0.811        | 0.857        | 0.829                 |
| 11   | <b>0.972</b> | 0.971        | <b>0.892</b> | <b>0.930</b> | <b>0.907</b>          |
| 12   | 0.961        | 0.969        | 0.838        | 0.899        | 0.861                 |
| 13   | 0.950        | 0.938        | 0.811        | 0.870        | 0.833                 |
| 14   | 0.939        | 0.933        | 0.757        | 0.836        | 0.787                 |
| 15   | <u>0.861</u> | 0.875        | <u>0.378</u> | <u>0.528</u> | <u>0.427</u>          |
| 16   | 0.950        | 0.938        | 0.811        | 0.870        | 0.833                 |
| 17   | 0.917        | 0.867        | 0.703        | 0.776        | 0.730                 |
| 18   | 0.939        | 0.906        | 0.784        | 0.841        | 0.806                 |
| 19   | 0.917        | 0.867        | 0.703        | 0.776        | 0.730                 |
| 20   | 0.872        | <u>0.750</u> | 0.568        | 0.646        | 0.597                 |
| 21   | 0.944        | 0.966        | 0.757        | 0.848        | 0.791                 |
| 22   | 0.889        | 0.774        | 0.649        | 0.706        | 0.670                 |
| 23   | 0.961        | 0.941        | 0.865        | 0.901        | 0.879                 |

## CHAPTER 4

### COMBINING SENSOR DATA USING CLASSICAL ENSEMBLES

The question our study aims to answer is *how to combine* the passive seismic data streams from each of the  $n$  sensors (geophones) in the network for best performance on the learning task. We explore two options.

- Option 1)** Use all  $n$  data streams to train one classification ensemble. Option 1 allows us to use existing ensemble methods such as Bagging, Boosting, and Random Forest.
  
- Option 2)** Use the data from each sensor to train a classifier for that sensor, resulting in  $n$  distinct classifiers. Option 2 could produce informative visualizations on the location of anomalies. That is, Option 2 might provide a map of sensor predictions that denotes the likelihood an anomaly is happening near a particular sensor.

In this chapter, we explore Option 1 and present the results of combining the data from all sensors for creating one ensemble; Chapter 5 then explores Option 2. Option 1, depicted in Figure 4.1, relies on sampling strategies for creating diversity in the base classifiers constituting the ensemble. We expect this approach will have improved performance over Option 2 since this approach uses all available data; however, Option 2 might locate anomalies (Chapter 5).

The ensemble will consist of multiple base classification models. For instance, although there are often hundreds or thousands of decision trees in a Random Forest, we treat the group of classifiers as one single model, the ensemble, and obtain one prediction for each observation.

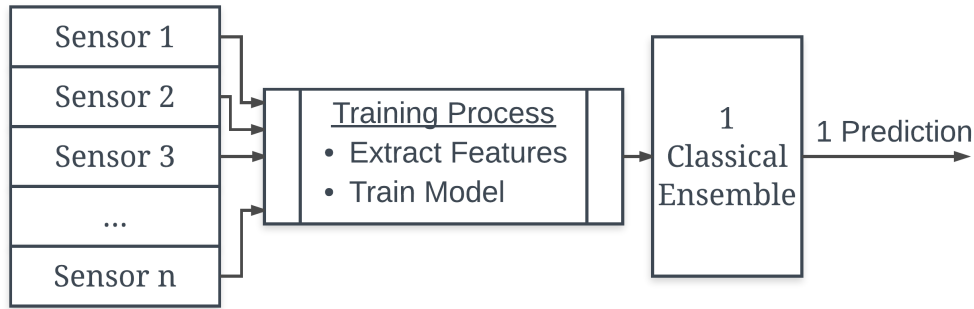


Figure 4.1: Diagram showing how one single ensemble trains using the data from  $n$  sensors (Option 1). Using this approach, we seek to improve upon the results of previous work that considered the data from only one sensor. Furthermore, this option provides one prediction about the state of the levee as compared to Option 2, which gives us the same number of predictions as sensors in the network.

#### 4.1 Experimental Setup

To empirically evaluate the performance of ensemble methods in our EDL application, we used three classes available in the `scikit-learn` framework - `BaggingClassifier`, `AdaBoostClassifier`, and `RandomForestClassifier`. As discussed in Chapter 2.1.2, Random Forest classifiers can only use decision trees as base classifiers. Bagging, however, can use any classification algorithm and AdaBoost only requires that the base classifier can output a probability class label.<sup>5</sup> We experimented with multiple base classifiers for the `BaggingClassifier` and `AdaBoostClassifier`: the `DecisionTreeClassifier`, `GaussianNB`,<sup>6</sup> `LogisticRegression`, and `SVC`<sup>7</sup> classifiers in `scikit-learn`.

We used the passive seismic data from all 23 sensors and class labels from the USBR piping experiment described in Chapter 3.1 for training the supervised learning ensembles. We again used level three Haar wavelet transforms to reduce noise in the passive seismic data be-

<sup>5</sup>A probability class label is the probability, in the range  $[0, 1]$ , that an observation belongs to a certain class label. This differs from predicted class labels which are discrete values (e.g., 0 or 1) and not probabilities.

<sup>6</sup>The `GaussianNB` class implements a Naive Bayes method that approaches classification using conditional probability as defined by Bayes' Theorem; the `GaussianNB` assumes that all features are independent and of Gaussian distributions [19].

<sup>7</sup>The `SVC` class implements a two-class support vector machine with a radial basis function (RBF) kernel by default.

fore binning the data into three second windows and extracting eight features (see Table 3.1). This process yielded 720 observations with 184 features describing each observation.

When using the `SVC` class as the base classifier for the Boosting and Bagging ensembles, we employed the same feature scaling pre-processing step used for the support vector machines described in Chapter 3.2 to ensure the values for each feature were of the same magnitude. Unlike the process described in 3.2, we did not select an optimal subset of features as a pre-processing step; instead the ensemble methods choose feature subsets for their base classifiers to induce diversity. The feature scaling was only performed when using the support vector machines as base classifiers. The other classifiers used with the ensembles do not require feature scaling because they are not as sensitive to variability in the ranges of features used as input.

## 4.2 Results and Discussion

In training the ensemble classifiers, we employed stratified 5-fold, cross-validation to tune the hyper-parameters of each classifier using the `GridSearchCV` class in `scikit-learn`. For all ensembles, we searched for the best number of base classifiers among the values 10, 25, 50, 100, 250, and 500. Other parameter settings searched during the cross-validation step depended on the type of ensemble. Chapters 4.2.1 through 4.2.3 discuss the results of cross-validation to find the hyper-parameter settings expected to generalize well with unseen data. Chapter 4.2.4 discusses the performance metrics of each ensemble evaluated on the observations in the test set.

### 4.2.1 Bagging Ensembles

For the `BaggingClassifier` ensembles, we used cross-validation to select the variation of Bagging to use for our task. The `scikit-learn` implementation takes boolean arguments for whether to sample observations in the training set with or without replacement (e.g., Bagging [16] or Pasting [35]) and whether to sample subsets of the features (e.g., Random Subspaces [36]) or whether to sample subsets of both features and observations (e.g., Random

Patches [37]).

For each variation of Bagging, we also searched for the optimal maximum fraction of samples and features considered when drawn with replacement from among the values 0.25, 0.5, 0.75, 1.0 to see whether the ensemble performed better when the base classifiers were provided more or less information. Table 4.1 shows the results of grid search cross-validation to find good hyper-parameters for the Bagging ensembles using the decision tree (DT), Gaussian naive Bayes (GNB), logistic regression (LR), and support vector machine (SVM) base classifiers.

Table 4.1: Hyper-parameter values selected through grid search cross-validation using `GridSearchCV` class for the `BaggingClassifier` and  $F_1$  score as the cross-validation metric.

| Base Classifier | # of Classifiers | Bootstrap    |          | Fraction per Training Set |          |
|-----------------|------------------|--------------|----------|---------------------------|----------|
|                 |                  | Observations | Features | Samples                   | Features |
| DT              | 500              | True         | True     | 1.0                       | 0.75     |
| GNB             | 500              | True         | True     | 0.75                      | 0.25     |
| LR              | 250              | False        | True     | -                         | 0.5      |
| SVM             | 25               | False        | True     | -                         | 0.25     |

From Table 4.1, we see that DT and GNB based ensembles performed better in cross validation with more classifiers than bootstrap observations and features. LR and SVM ensembles, on the other hand, performed better with only bootstrapping features. The SVM based ensemble also only used 25 classifiers, the second lowest value searched with `GridSearchCV`.<sup>8</sup> Using fewer classifiers and opting not to resample observations suggests the SVMs are hungry for more observations to increase variability in the models when compared to the other base classifiers. Furthermore, the SVM, LR, and GNB based ensembles performed best when considering a smaller fraction of the available features for each observation. This result matches our expectations because the SVM, LR, and GNB models are linear models, which are more subject to the curse of dimensionality<sup>9</sup> than tree based

<sup>8</sup>Recall that we searched among the values 10, 25, 50, 100, 250, and 500 for the optimal number of base classifiers in the ensemble.

<sup>9</sup>In machine learning, the curse of dimensionality is as follows: With fixed number of observations, increasing the number of features can make the learning task more difficult.

ensembles.

### 4.2.2 Boosting Ensembles

For the AdaBoost ensembles, we varied the learning rate to be 0.1, 0.25, 0.5, 0.75, or 1. Learning rate is a coefficient controlling the contribution of each new classifier to the entire ensemble: a lower learning rate implies a base classifier added to the ensemble later in the fitting process has less impact than an earlier base classifier on the ensemble’s predictions. Table 4.2 shows the hyper-parameter values selected for the AdaBoost ensembles.

Table 4.2: Hyper-parameter values selected through grid search cross-validation using the `GridSearchCV` class with the `AdaBoostClassifier` as the base classifier and  $F_1$  score as the cross-validation metric.

| Base Classifier | # of Classifiers | Learning Rate |
|-----------------|------------------|---------------|
| DT              | 10               | 0.1           |
| GNB             | 25               | 0.1           |
| LR              | 10               | 1             |
| SVM             | 25               | 0.5           |

Regardless of the base classifier used, the best number of classifiers found through cross-validation was either 10 or 25, which are small ensembles when compared to the best number of classifiers found for the DT, GNB, and LR Bagging ensembles. This result implies that AdaBoost performs worse as more classifiers join the ensemble, which suggests AdaBoost suffers from focusing each new classifier on the examples most difficult to predict in our data set (e.g., the anomalies) rather than tightly fitting the normal data. Similarly, the decision tree and Gaussian Naive Bayes based AdaBoost ensembles found low learning rates perform better, which supports the claim that adding more classifiers to a Boosting ensemble will not improve performance on our learning task.

### 4.2.3 Random Forest Ensembles

For the Random Forest ensembles, we determined whether the Gini impurity criterion or entropy information gain was the appropriate measure for quality of splits at a single node in

the decision trees via cross-validation. The quality of splits is used in the fitting process for determining the best feature to use in partitioning the data set into the appropriate classes; a feature with a higher quality split will more clearly divide the classes of observations. The Gini criterion, contrary to entropy gain, assumes the attributes of the data set are continuous and not categorical [38]. Also, in the cross-validation search for optimal parameters, we determined whether to bootstrap training observations, as in Bagging. Lastly, to force diversity in the decision functions learned by each tree, we varied the maximum number of features considered for every split in each decision tree to be the total number, the square root, or the base 2 logarithm of the number of features.

The cross-validation search for optimal hyper-parameters for the Random Forest ensembles did not yield consistent results for different random seeds. The differences in cross-validation metrics were of the same magnitude as their standard deviations for the hyper-parameter configurations searched. Thus, the hyper-parameters reported here may not be optimal. Unlike Bagging and Boosting, we observed that different hyper-parameter configurations for the Random Forest ensembles performed similarly on both the cross-validation and test sets.

Because the hyper-parameter configurations searched performed similarly, we only report the performance of a Random Forest consisting of 25 decision trees. The trees do not bootstrap observations, unlike Bagging Decision Trees, and only consider  $\log_2$  of the total number of features. We did find that the Gini criterion was better than the entropy information gain for measuring the quality of a split in training the decision trees. Since the attributes of our data set are continuous, the fact that the Gini criterion performed better matched our expectations.

Table 4.3 shows the resulting feature importances of the Random Forest trained using the hyper-parameter settings specified previously. A higher importance rating implies earlier splits in the forest's trees use the feature, meaning the feature is responsible for directing the final prediction of more examples. RMS energy (defined in Table 3.1) appears to be the most

important feature extracted from each sensor; furthermore, RMS Energy from sensors 9, 10, 8, and 4 appear highest on the list. Some features, such as Zerocross, from some sensors (e.g., 2, 11) are also comparable to the RMS Energy of other sensors (e.g., 13, 16) in terms of importance rating; however, we believe RMS Energy strongly indicates the state of the EDL structure due to the following intuition: a structure undergoing erosion will see highly energetic events (e.g., the sliding of slopes or other removal of earthen materials) whereas a healthy structure should remain relatively inert.

Table 4.3: The top twenty features used for splitting observations in decision trees of the `RandomForestClassifier`.

| Sensor Number | Feature        | Importance Rating |
|---------------|----------------|-------------------|
| 9             | RMS Energy     | 0.050966          |
| 10            | RMS Energy     | 0.049973          |
| 8             | RMS Energy     | 0.048420          |
| 4             | RMS Energy     | 0.044419          |
| 2             | RMS Energy     | 0.037069          |
| 12            | RMS Energy     | 0.034084          |
| 18            | RMS Energy     | 0.033311          |
| 5             | RMS Energy     | 0.032119          |
| 22            | RMS Energy     | 0.030874          |
| 11            | RMS Energy     | 0.029086          |
| 17            | Spread         | 0.029075          |
| 2             | Zerocross rate | 0.024963          |
| 13            | RMS Energy     | 0.024489          |
| 16            | RMS Energy     | 0.024007          |
| 11            | Zerocross rate | 0.022395          |
| 19            | Zerocross rate | 0.022070          |
| 4             | Skewness       | 0.020701          |
| 20            | Kurtosis       | 0.020513          |
| 1             | Rolloff        | 0.020389          |
| 17            | Rolloff        | 0.019644          |

#### 4.2.4 Generalization Abilities of Ensembles

Table 4.4 summarizes the performance of the different ensemble algorithms trained on passive seismic data from all 23 sensors in the USBR piping experiment. In this table, the Bagging and AdaBoost ensembles with SVMs exhibit the best  $F_1$  scores. Bagging with SVMs

also show the best accuracy and precision of all the ensembles, while AdaBoost with SVMs shows the best  $F_\beta$  ( $\beta = 2$ ) score. While these two SVM models show top performance in four of the five performance metrics described in Table 2.1 (including  $F_1$  score - a key metric of comparison in our study), Random Forest, Bagging with decision trees, and Bagging with logistic regression all show  $F_1$  scores above 90%.

Table 4.4: Performance metrics for each ensemble trained using passive seismic data from all 23 sensors in the USBR piping experiment. The models trained using the best parameters found for each ensemble through the grid search cross-validation process. The best result in each column is in bold. The worst value in each column is underlined. Average and standard deviations are reported for each model using 10 iterations of training and testing with different partitions of the data set.

| Ensemble       | Accuracy                            | Precision                           | Recall                              | $F_1$                               | $F_\beta$ ( $\beta = 2$ )           |
|----------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Bagging - DT   | 0.976 $\pm$ 0.013                   | 0.951 $\pm$ 0.043                   | 0.924 $\pm$ 0.045                   | 0.936 $\pm$ 0.032                   | 0.929 $\pm$ 0.037                   |
| Bagging - GNB  | 0.952 $\pm$ 0.019                   | 0.828 $\pm$ 0.066                   | 0.957 $\pm$ 0.043                   | 0.886 $\pm$ 0.044                   | 0.927 $\pm$ 0.036                   |
| Bagging - LR   | 0.973 $\pm$ 0.009                   | 0.951 $\pm$ 0.033                   | 0.905 $\pm$ 0.041                   | 0.927 $\pm$ 0.025                   | 0.914 $\pm$ 0.033                   |
| Bagging - SVM  | <b>0.979 <math>\pm</math> 0.006</b> | <b>0.964 <math>\pm</math> 0.028</b> | 0.927 $\pm$ 0.036                   | 0.944 $\pm$ 0.015                   | 0.933 $\pm$ 0.026                   |
| AdaBoost - DT  | 0.952 $\pm$ 0.025                   | 0.899 $\pm$ 0.046                   | <u>0.850 <math>\pm</math> 0.078</u> | 0.873 $\pm$ 0.058                   | <u>0.859 <math>\pm</math> 0.069</u> |
| AdaBoost - GNB | <u>0.947 <math>\pm</math> 0.024</u> | <u>0.822 <math>\pm</math> 0.070</u> | <b>0.962 <math>\pm</math> 0.049</b> | <u>0.870 <math>\pm</math> 0.058</u> | 0.903 $\pm$ 0.052                   |
| AdaBoost - LR  | 0.962 $\pm$ 0.013                   | 0.925 $\pm$ 0.040                   | 0.875 $\pm$ 0.067                   | 0.897 $\pm$ 0.037                   | 0.883 $\pm$ 0.054                   |
| AdaBoost - SVM | 0.972 $\pm$ 0.007                   | 0.951 $\pm$ 0.039                   | 0.941 $\pm$ 0.035                   | <b>0.945 <math>\pm</math> 0.017</b> | <b>0.943 <math>\pm</math> 0.024</b> |
| Random Forest  | 0.970 $\pm$ 0.011                   | 0.955 $\pm$ 0.041                   | 0.888 $\pm$ 0.059                   | 0.919 $\pm$ 0.03                    | 0.899 $\pm$ 0.046                   |

Peculiarities of the training data set would influence the results of hyper-parameter selection through cross-validation for the Random Forest, which suggests the selected Random Forest model might overfit the data [39]. Cawley and Talbot propose methods for relieving this possible over-fitting in model selection [39]. However, the hyper-parameter values found during the model selection process can still show good generalization even when optimal hyper-parameters are not found [39]. Future work could explore either (1) investing in a larger, more robust data set to use for model selection or (2) attempting to tune the hyper-parameters further using methods in [39] with the USBR piping data set.

For the DT, GNB, and LR base classifiers, the  $F_1$  scores evaluated on the test sets for the ensembles indicate Boosting’s performance with these base classifiers is inferior to the Bagging and Random Forest ensembles. We expected Boosting to perform worse than the

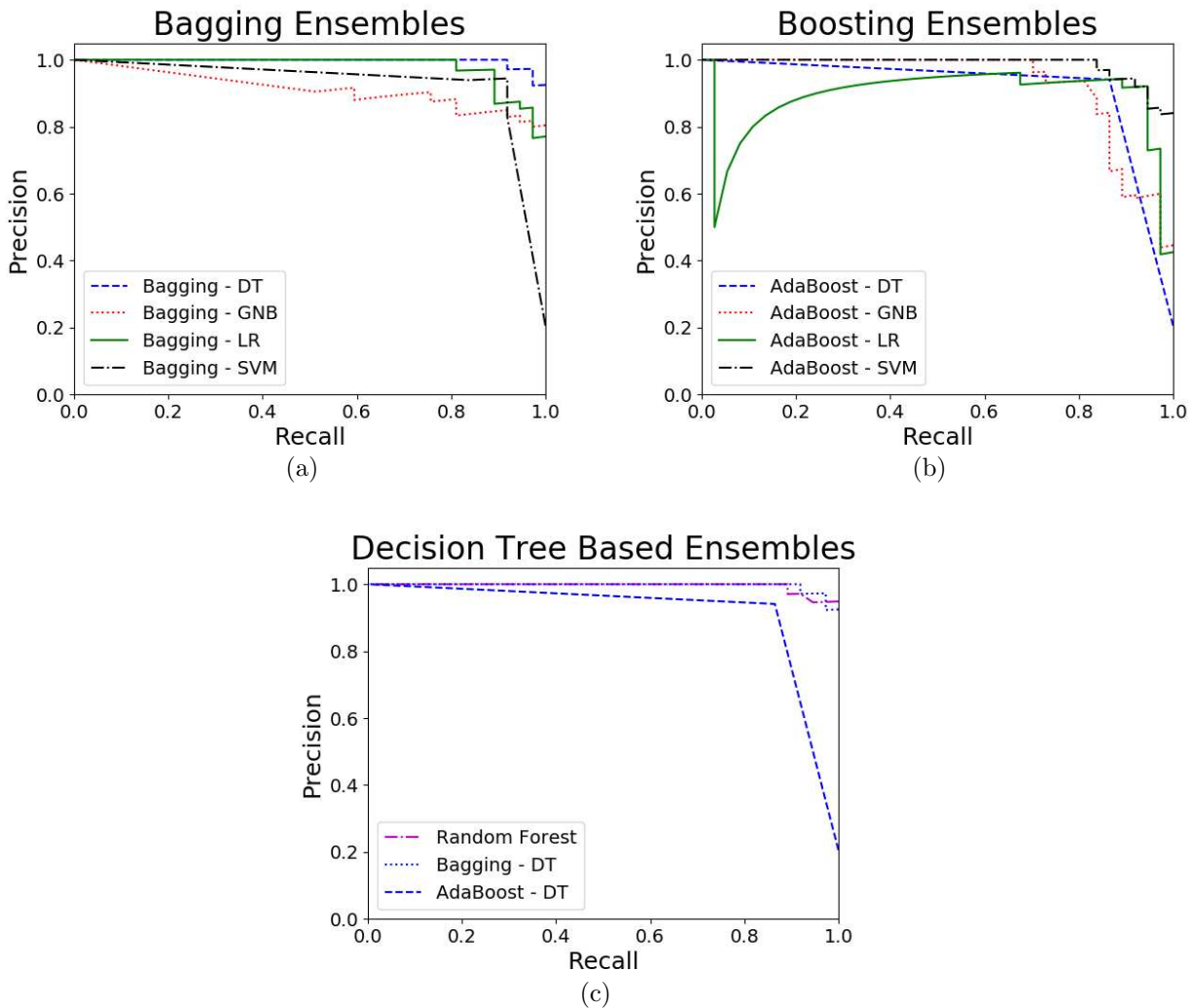


Figure 4.2: Precision-Recall curves showing the trade off between different probability thresholds for predicting positive class labels. We note that Precision-Recall curves are often considered more appropriate than the other, commonly used, Receiver Operating Characteristic (ROC) curves when analyzing performance on learning tasks with imbalanced data sets [40]. ROC considers recall against the probability of a false alarm, whereas precision-recall curves consider recall against the value of positive predictions.

other ensembles, as Boosting is not well suited for learning tasks where the data contains significant outliers. As discussed in Chapter 2.1.2, the base classifiers in Boosting ensembles will focus on fitting the outlier data while anomaly detection methods typically focus on tightly fitting the normal data. Furthermore, we also expected the GNB base classifier to perform poorly in each ensemble because the GNB algorithm makes a false assumption about

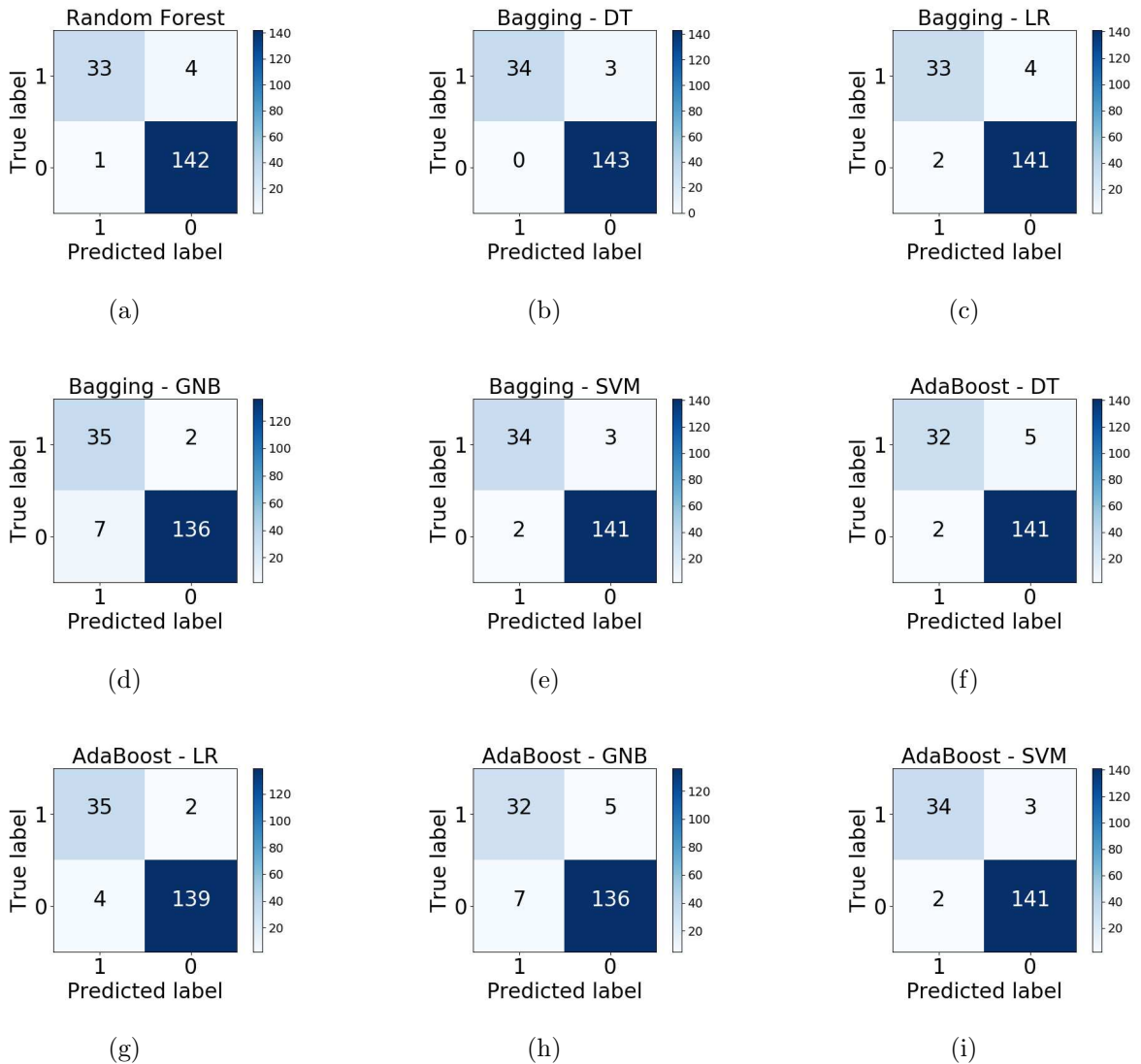


Figure 4.3: Confusion matrices for all ensembles showing counts of correct and incorrect classifications on the test set by type. Darker shading means that quadrant contained a larger fraction of the test set; less shading in the off-diagonal, or error, quadrants of the matrix indicates better performance.

our data set, i.e., all attributes of the data set are of independent Gaussian distributions. Table 4.4 reflects this expectation; GNB is the worst performer, in terms of  $F_1$  scores, of the four base classifiers in both Bagging and Boosting. We experimented with Boosting to provide a comparison for the other methods and our results reflect our expectations except in the case of the SVM base classifier. For the SVM base classifier, the Boosting ensemble

was able to improve recall over the Bagging SVMs at the expense of precision. This trade-off explains the similarity in  $F_1$  scores for SVMs in Boosting and Bagging while also accounting for the Boosting SVMs' superior  $F_\beta$  ( $\beta = 2$ ) score, which favors recall more than precision.

As discussed in Chapter 2, there are different costs associated with false negative and false positive errors. A dam operator in a real deployment might prefer an increase in false positives (false alarms) if the rate of false negatives (erosion events that go undetected) could decrease. Figure 4.2 shows the relationship between precision and recall as the threshold for predicting a positive label varies for the ensemble methods we evaluated. Figure 4.2 provides insight that could help dam operators decide an appropriate threshold for alerts depending on their operating costs. The Precision-Recall curve for a good classifier will reside in the upper-right-hand corner [40], meaning the classifier rarely raises false alarms (high precision) while detecting most of the anomalous events (high recall). Figure 4.2(b) shows that Boosting with SVMs, although achieving the highest  $F$ -scores, must sacrifice high precision in order to obtain high recall; Figure 4.2(c) shows, however, that the Random Forest and Bagging DT ensembles do not sacrifice precision as quickly in order to obtain high recall. Thus, Random Forest and Bagging DT ensembles should still be considered for use in real deployments despite their  $F$ -scores not being the highest in Table 4.4.

Figure 4.3 shows the confusion matrices for the predictions of each ensemble method on the test set. True negatives, i.e., anomalous observations predicted as normal, are the more expensive error for our application; Figure 4.3 shows that Bagging with GNB and Boosting with LR minimize the number of such errors. This minimization, however, comes with more false positive errors when compared to all other methods except Boosting with GNB. A dam operator would need to weigh this tradeoff when deciding on a model to use.

The results presented in this section demonstrate that utilizing the data from more than one sensor in the USBR piping experiment can improve performance in terms of  $F_1$  score on the classification task. Boosting SVMs demonstrate a 1.5% increase in  $F_1$  score over the best performing support vector machine in Table 3.2 (which used data from only one sensor).

More importantly, Boosting SVMs demonstrate a 41.7% increase in  $F_1$  score over the worst performer in Table 3.2. These results show that employing the data from all sensors in a structure should improve the reliability of predictions from machine learning methods that detect erosion events in earth dams and levees.

### 4.3 Analysis of Computational Resource Requirements

To understand the requirements for computational time and space resources, we measured the memory usage and user time required to fit the training set and predict class labels for the test set for each of the ensembles reported in this chapter. We measured the time requirements as the difference between two calls to `time.time()`, which returns the time in seconds rounded to three decimals of precision. We measured the space/memory requirements using the `memory_profiler` package in Python, which reports the total memory usage of the Python interpreter as each command executes. Computing the difference in the memory usage of the interpreter for each command allows one to understand the size of objects throughout their life-cycle. The `memory_profiler` also provides an API for monitoring the memory usage of a given function call by sampling the total memory allocated to the Python process throughout the function’s execution.

Table 4.5 presents every ensemble model’s average time and memory requirements for both fitting the training set and predicting the labels for the test sets over 100 repetitions. The time to make new predictions is the most important metric in the table, since we want to provide continuous monitoring systems that can classify new data in real-time. On the other hand, it is acceptable for a classification model to train for days (or even weeks) before deployment in a real system. All measurements are the averages of 100 repetitions of fitting each model and obtaining that model’s predictions for the test set. Each repetition ran on a Linux desktop machine running Python 3.5 and MATLAB R2017b with 8 Gigabytes of RAM and an Intel i7-2600k processor clocked at 3.4 Gigahertz.

The memory usage of the Bagging models, as measured by the `memory_profiler` tool with results shown in Table 4.5, is higher for the prediction process than the memory usage

Table 4.5: The average time and memory to fit training set (of 540 observations) and to predict class labels for the test set (of 180 observations) with 100 repetitions of each ensemble in Python (times in seconds; memory in Megabytes).

| Ensemble       | Time to Fit | Time to Predict | Memory to Fit | Memory to Predict |
|----------------|-------------|-----------------|---------------|-------------------|
| Bagging - DT   | 3.933       | 0.348           | 1.004         | 1.031             |
| Bagging - GNB  | 1.299       | 0.469           | 0.496         | 0.616             |
| Bagging - LR   | 6.099       | 0.568           | 0.135         | 0.389             |
| Bagging - SVM  | 0.648       | 0.329           | 5.527         | 0.051             |
| AdaBoost - DT  | 0.203       | 0.077           | 0.0           | 0.0               |
| AdaBoost - GNB | 0.375       | 0.469           | 1.405         | 0.006             |
| AdaBoost - LR  | 0.846       | 0.072           | 0.756         | 0.0               |
| AdaBoost - SVM | 12.797      | 0.537           | 16.518        | 0.0               |
| Random Forest  | 0.524       | 0.111           | 0.93          | 0.0               |

for the AdaBoost and Random Forest models. This result is likely attributed to the fact that Bagging models can run in parallel, so more memory must be allocated to keep track of the work done in separate processes. The zero values reported for memory usage in four models imply the `memory_profiler` did not detect a significant change in the memory allocated to the Python interpreter while obtaining predictions. One can also see that the SVM models (both Bagging and Boosting) required the most memory during the fitting process, which is likely due to the fact that SVM models require computing the results of kernel functions for the training sets and are generally more complex models to fit than Decision Trees, Logistic Regression, and Gaussian Naive Bayes.

We were not able to find a tool for profiling memory usage of each MATLAB pre-processing step in our system. However, we do know that one SEG-Y file containing 30 seconds of raw signal for each of the 23 sensors is 1.6 Megabytes. We also observed that the size of a file containing the eight extracted features for one sensor’s data for 10 observations is less than 1 kilobyte. The size of the files storing the features for one sensor grows semi-linearly with each added observation (e.g., 100 observations requires about 8 kilobytes and 1000 observations requires about 80 kilobytes). *In summary, the memory requirements for our continuous monitoring system are easily satisfied by modern computing hardware available*

*to average consumers.*

We also profiled the time requirements for the pre-processing portion of our machine learning pipeline in MATLAB. This pre-processing includes loading SEG-Y<sup>10</sup> files of the raw signal for all 23 sensors, completing Haar Wavelet denoising for each sensor’s signal, decomposing each signal into 3 second frames, and extracting the eight descriptors of every frame. Across 72 SEG-Y files, the average load time for a single file of data was approximately 0.146 seconds.<sup>11</sup> Furthermore, the time necessary for denoising the 30 seconds of raw data from one sensor is approximately 0.012 seconds. Decomposing 10 frames from the denoised signal and extracting 8 features for each of those frames requires approximately 0.615 seconds. Thus, it requires about 0.773 seconds *per sensor* to complete the pre-processing (including loading the signal, denoising the signal, frame decomposition, and feature extraction) of 10 observations.

As discussed previously, the time to fit a model does not have to satisfy any real-time constraints whereas predictions must be made in real-time. As shown by the average time to predict class labels for all 180 observations in the test set (see Table 4.5), these machine learning models are capable of providing real-time, continuous monitoring of Earth dams and levees. Recording a new SEG-Y file with 10 observations requires 30 seconds, while the average time required to load the file, complete Haar wavelet denoising, and extract the features for 10 three second frames from all 23 sensors is about 17.779 seconds (23 sensors  $\times$  0.773 seconds for each) in MATLAB. *In summary, our systems would be able to process each new SEG-Y file in its entirety while the next is being recorded.*

---

<sup>10</sup>SEG-Y is a standard file format for geophysical data.

<sup>11</sup>A single file of SEG-Y data has 30 seconds of passive seismic data from all 23 sensors.

CHAPTER 5  
COMBINING SENSOR DATA WITH DISTINCT CLASSIFIERS

Option 1, explored in Chapter 4, combined the features from each sensor into the same observation so one classifier considered the data from all the sensors. Option 2, explored in this chapter, trains 23 different classifiers by maintaining separation of the features for each sensor. Having one model per sensor allows us the opportunity to associate the prediction of a model with that sensor’s location. We hypothesize that distinct models will detect anomalies more often or with more confidence (i.e., probability predictions close to 1) for the sensors closer to the piping anomaly. If we are able to spatially map the anomalous events using Option 2, then we can direct human experts to investigate a particular portion of the dam or levee that may be failing. Figure 5.1 depicts Option 2.

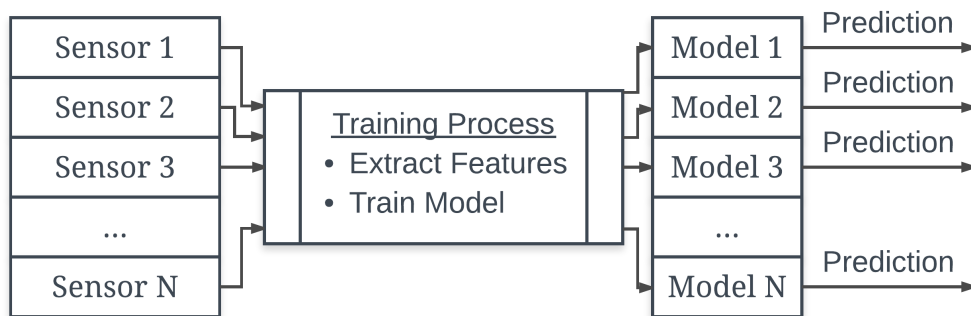


Figure 5.1: Diagram showing how distinct models will train using the data from each sensor (Option 2). We hypothesize that maintaining separation among the predictions of the models will allow us to create visualizations on where an anomaly is taking place in the EDL. In this approach, the prediction of each model represents the likelihood an anomaly is occurring near the corresponding sensor.

To help clarify the difference between Option 1 (Chapter 4) and Option 2 (Chapter 5), imagine you are a professor deciding whether to make a particular assignment a team project or an individual assignment. By making the assignment a team project, you are allowing the

students to collaborate on the task and typically would expect better work to result from the collaboration. On the other hand, if you choose to make the assignment an individual task, you expect to learn which students understand the material better than others and, therefore, which students need more attention to properly learn the material. The expectations behind these two cases correspond to Option 1 (classical ensembles) and Option 2 (distinct models) respectively.

### 5.1 Distinct Supervised Support Vector Machines

In this work, we explore whether distinct machine learning models for each sensor can provide informative visualizations of erosion in an EDL structure. We first trained supervised, two-class SVMs on the data from each sensor (see Chapter 3). We then plotted the predictions of each SVM for all 180 samples in the test set using the binary class labels output by the `predict` functions of each trained SVM. When a sensor’s SVM model reports a negative class label (i.e., no anomaly detected), we color the circle corresponding to the sensor’s location as green. When a sensor’s SVM model reports a positive class label (i.e., anomaly detected), we color the circle red. Figure 5.2 displays a sample of the resulting visualizations. We show four example visualizations from the 180 total as the four samples reflect the majority of visualizations observed. Figure 5.2(a) covers the cases where a normal observation has a few models predicting an anomaly. Figure 5.2(c) covers the cases where many models predict an anomalous observation to be normal. Figure 5.2(b) and Figure 5.2(d) cover the cases where all models predict the correct class label. We note the most common case in the 180 visualizations was a normal observation being predicted normal by all models. Figure 5.2 does not support our initial hypothesis that there would be a clear pattern of sensors closer to the right side of the structure (near the piping erosion) detecting the anomalies while sensors closer to the left side of the structure do not.

We then evaluated the probability prediction (i.e., the probability that the passive seismic data indicates an anomaly) from each sensor’s SVM model for all 180 samples in the test set. Figure 5.3 shows four example visualizations from our analysis. In this figure, we

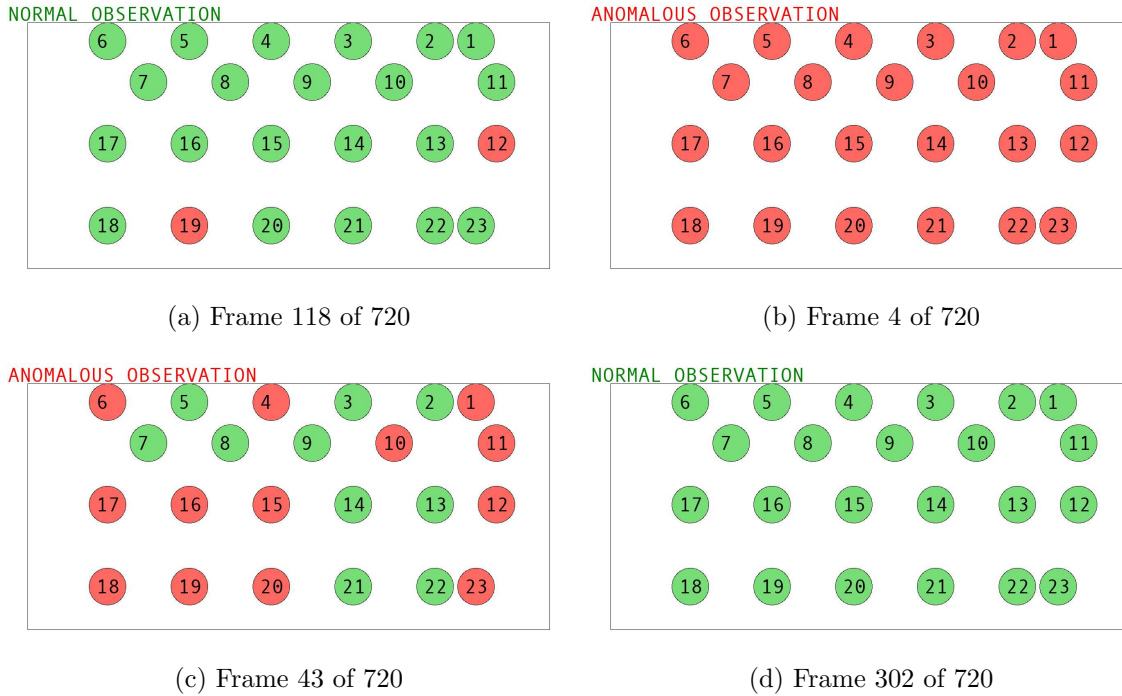


Figure 5.2: Sample visualizations from the predictions of supervised SVM models. Each prediction uses the data from the corresponding geophone sensor. The text over the top left of each subfigure denotes the ground truth label of the given 3 second frame. A red circle means the SVM predicts the observation is anomalous, while a green circle means the SVM predicts the observation is normal.

plot the probability predictions rather than the binary class labels in order to gain some insight into which sensor’s models are more confident in the class label output than others. Figure 5.3 provides this data for the same observations presented in Figure 5.2. For example, Figure 5.2(b) shows that all models are predicting the observation at that point in time as an anomaly. Figure 5.3(b) shows that, for the same observation, model 9 is not as confident in its prediction as the other models. Specifically, the model for sensor 9 predicts an anomaly with only 0.65 probability; other models, such as the model for sensor 10, predict the anomalous label with 0.99 probability. Figures 5.2 and 5.3 show that visualizations for binary class labels showing homogeneous labels for each sensor can be misleading, i.e., there are still subtle differences in the decision functions learned by each SVM. However, these subtle differences are not enough to show clear patterns on where an erosion event is located. In

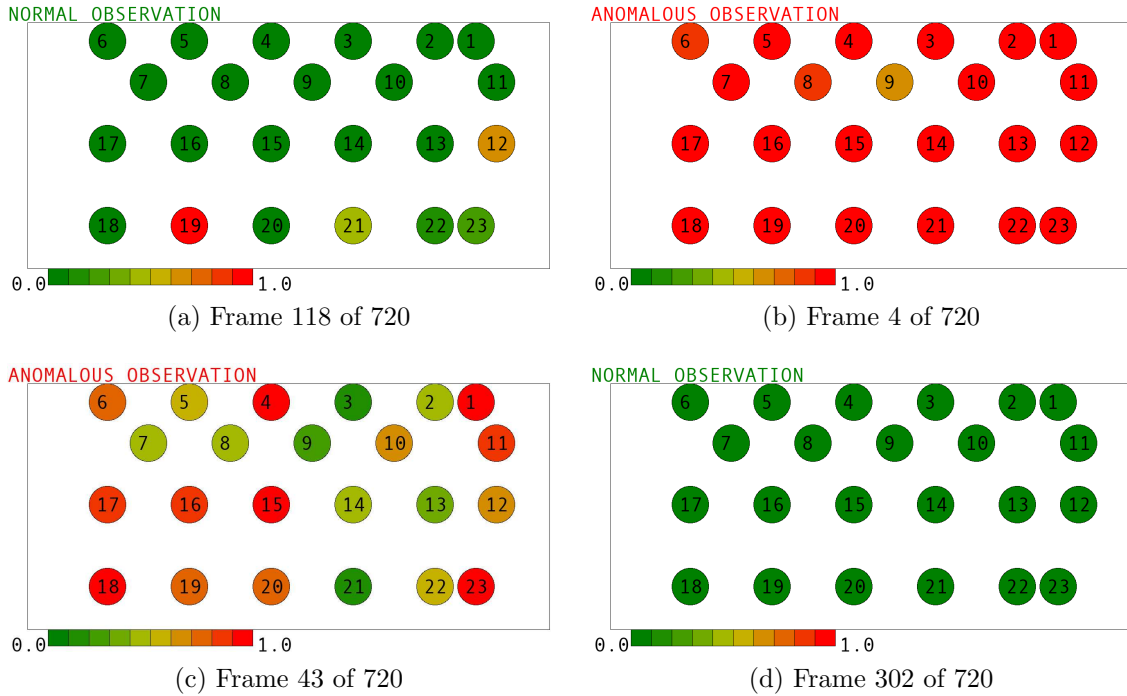


Figure 5.3: Sample visualizations from the probability predictions of supervised SVM models, where each prediction uses the data from the corresponding geophone sensor. The probability ranges from 0.0 (no anomaly detected) to 1.0 (anomaly detected).

some cases, all models predict the correct class label. In other cases, the model predictions differ, but there is no pattern of models closer to the known anomaly predicting anomaly more often or more confidently.

As discussed previously, all observations are labeled anomaly if there exists an anomaly anywhere in the structure. Thus, an open question for Option 2 follows: when using supervised models, how do we create labels for each individual sensor’s data? We expect data collected from sensors farther away from erosion events to be less likely to represent the anomaly. If we use a supervised learning approach, then training and test labels for our observations must reflect this expectation. In other words, how close must a sensor be to an event to detect that event? A semi-supervised approach (Chapter 5.2) is a solution to this dilemma, because we only train with observations that we know contain no erosion events/anomalies anywhere in the structure.

## 5.2 Distinct Semi-Supervised One-Class Support Vector Machines

We discussed the difficulty in creating a set of ground-truth labels for each of the sensors depending on their location and distance to the known anomalies. As an alternative, we discuss our investigation into semi-supervised learning and one-class support vector machines (OCSVMs) for each of the sensors in this section. These OCSVMs only used the data assumed to be free from erosion events (i.e., the OCSVMs were only trained on normal observations). Before training the OCSVMs, we moved all anomalous observations in the training set described in Chapter 3.2 to the test set. Thus, the one-class support vector machines trained using 444 normal observations and then used 276 observations for validation and testing (143 normal and 133 anomalous). As in [2], we only used the RMS Energy and Zerocross features as inputs for the OCSVMs, as we know the OCSVM for sensor 11 performed better when only considering these two features (instead of all eight); see Table 3.1 for details.

In the `Scikit-Learn` framework, the one-class support vector machine model with an RBF kernel has hyper-parameters  $\nu$  (an upper bound on the fraction of training errors [19]) and  $\gamma$  (the coefficient for the kernel function [19]). To tune these hyper-parameters, we used holdout cross validation on the observations from sensor 11. That is, we split the 276 observations not used for training into two distinct sets: a validation set for choosing hyper-parameters and a test set for reporting generalization performance. The validation set contained 138 observations (77 normal and 61 anomalous); the test set also contained 138 observations (66 normal and 72 anomalous). We used a grid search to experiment with different values for the hyper-parameters  $\nu$  and  $\gamma$ . That is, we fit OCSVM models to the 444 normal training observations from sensor 11 and selected the  $\nu$  and  $\gamma$  which yielded the best  $F_1$  score on the holdout validation set.

We searched for the optimal  $\gamma$  among the values  $1.0e+x$ , where  $x$  is an integer on the range  $[-9, 3]$ ; we searched for the optimal  $\nu$  among the real values from 0 to 1 in 0.005 intervals. We chose the exponentially spaced range of possible  $\gamma$  values according to advice

in the `Scikit-Learn` user guide [19]. We chose the possible values of  $\nu$  because the value is restricted on the range  $(0, 1]$  and we decided 0.005 increments would provide a fine-grained search. From this search, we found the best performing OCSVM on the validation set used  $\gamma = 1.0e-7$  and  $\nu \approx 0.53$  for an  $F_1$  score of 0.987. The same model then obtained an  $F_1$  score of 0.962 on the test set observations, which suggests the OCSVM was able to fit the training and validation sets well and, more importantly, generalize to unseen observations. This OCSVM model performs comparably to the best OCSVM model presented by Fisher et. al. in [2], which obtained an  $F_1$  score of 0.95.

We ignore how well the OCSVMs perform using data from other sensors, because we are not confident the ground truth labels made for sensor 11 data are appropriate ground truth labels for the other sensors' data. Rather, we assume the chosen hyper-parameter values for the OCSVM of sensor 11 are appropriate for the other OCSVMs, and we evaluate whether the remaining 22 models can reliably fit the training data for the corresponding sensor.

After fitting each of the 23 OCSVM models to the 444 training observations from each sensor, we plot the predictions of all OCSVM models on the 276 observations not used in training the models. These plots (four of which are shown in Figure 5.4) allow us to see which sensors predict erosion events when we know there is erosion near sensor 11 and which sensors predict normal events when we know the dam is not actively eroding. With the two-class SVMs, the majority of the 180 visualizations were similar to Figure 5.2(d) where every model predicts normal observations as normal. With the OCSVMs, however, the majority of the 276 visualizations are similar to Figure 5.4(a) and Figure 5.4(d) where some or many of the models predict normal observations as anomalous. We also note that more OCSVMs agree that the anomalous observations are anomalies when compared to the two-class SVMs (i.e., compare Figure 5.4(c) to Figure 5.2(c)).

We now compare the sample visualizations of the semi-supervised OCSVM models (Figure 5.4) to the visualizations of the supervised SVMs on the same observations (Figure 5.2 and Figure 5.3). The semi-supervised models more often predict anomalies for observations

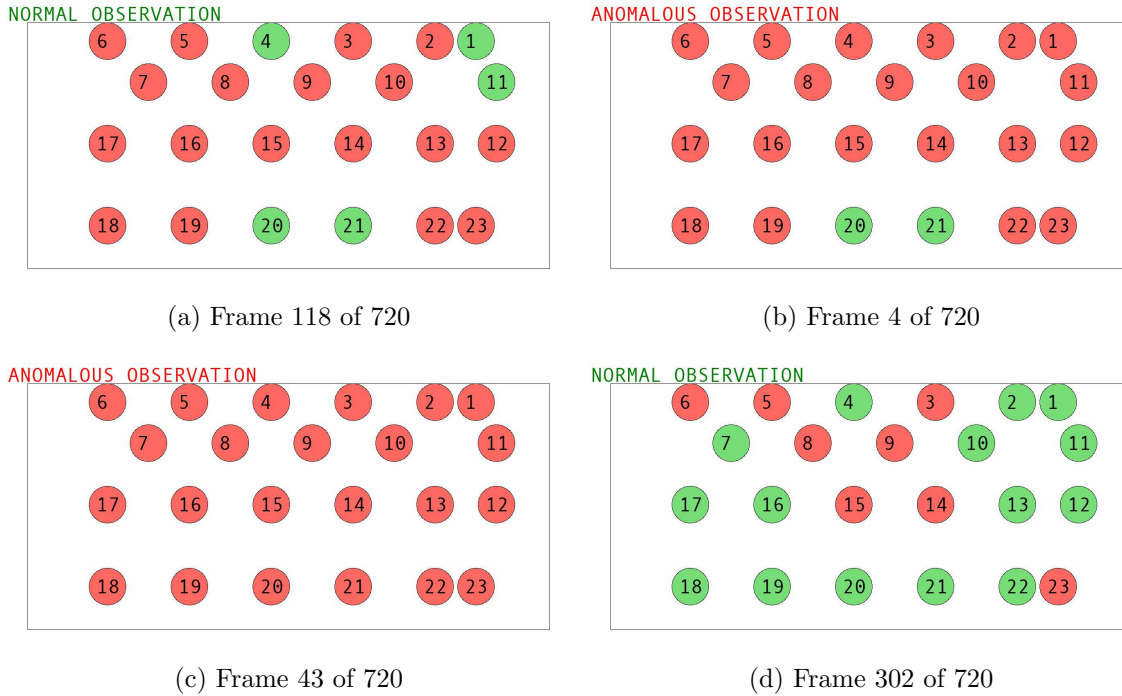


Figure 5.4: Sample visualizations from the predictions of semi-supervised OCSVM models, where each prediction uses the data from a distinct geophone sensor.

we believe do not contain any erosion in the structure. Almost all of the SVMs predict the normal observations in Figures 5.2 and 5.3, subfigures (a) and (d), to be normal. These normal observations in Figure 5.4, however, are not predicted normal as often. In fact, the majority of OCSVM models predict the normal observations in Figure 5.4(a) to be anomalous. This result suggests that there may be smaller erosion events taking place unseen elsewhere in the structure.

We conclude that neither the OCSVM nor SVM models trained on a per-sensor basis are able to clearly localize the erosion events in our small-scale embankment. In future work, we propose to investigate whether our localization methods might work at a larger-scale, i.e., perhaps the sensors in our embankment are simply too close together to make such visualizations. In other words, perhaps sensors spread out further will show clearer separations, in time and space, of normal and anomalous data. We are also curious if the wooden frame built to contain the small-scale USBR embankment could have spread

the seismic signal from the erosion event across the entire structure in ways that are not reflective of the real-world scenario.

## CHAPTER 6

### CONCLUSIONS AND FUTURE WORK

In this research, we first showed that the  $F_1$  score of an existing workflow for classifying erosion events in a laboratory embankment varied up to 40% depending on the particular sensor chosen as input to the workflow (Chapter 3). We then used classical ensemble techniques (Bagging, Boosting, and Random Forest) with various base classifiers that together considered features extracted from the signals of all 23 sensors in the embankment (Chapter 4). We expected that these ensembles would show more reliable performance than previous studies because they considered data from all sensors instead of only one. The results met our expectations; even the least well suited ensemble algorithm, Boosting with Gaussian Naive Bayes, had a better  $F_1$  score on the classification task (Chapter 4) than nine of the models that used data from only one sensor (Chapter 3). The best ensemble algorithm, Boosting with SVMs, outperformed all models that used data from only one sensor. We conclude that using the data from all sensors can improve a classifier's ability to separate erosion events from observations of normal passive seismic data.

We also trained one classifier for each sensor in the laboratory embankment with the expectation that the predictions from the classifiers could be associated with the sensor's proximity to the anomaly (Chapter 5). By associating predictions with locations, we hoped to see a localization of anomalous events in the embankment (e.g., sensors closer to the piping erosion would more strongly detect the anomalies). Unfortunately, our expectations for this approach did not match our results.

The previous chapters outline the main contributions of our study for this thesis. The goals in the following sections of this chapter outline future work.

## 6.1 Feature Engineering

We plan to further explore the effects of feature engineering on our results for this particular problem. One of the biggest challenges to machine learning applications is deciding how to represent the data used to solve the learning problem [14]. Previous studies on EDLs have made use of nine spectral features common in audio analysis [1–3]. It is possible that the performance of our machine learning models could improve if we augment these spectral features with interaction features and polynomial features [14]. Interaction features essentially try to capture informative relationships between multiple features. For instance, we could consider a feature such as `Zerocross`  $\times$  `Centroid` to capture interactions between `Zerocross` and `Centroid`, and these interactions may have useful patterns for the learner. We could also consider polynomial features such as the square of `Zerocross`; polynomial features might help compact normal observations into a smaller region in the feature space making the learning task easier.

We can also explore the interactions between features extracted from the signals of multiple sensors. For example, there could be interesting patterns in the interaction of `Spread_Sensor_1`  $\times$  `Kurtosis_Sensor_5` that could help improve the performance of the machine learning models. The choice of how to represent observations strongly influences what the models can learn [14].

Another feature engineering decision to explore is whether performance improves if we allow the extracted windows of seismic data to overlap. For an example of this idea, consider 100 seconds of passive seismic data. We need to convert this continuous data to discrete observations for our learning algorithms. One choice is to have 10 windows of 10 seconds each with no overlap. Another choice is to have 10 windows of 12 seconds each, which allows overlap. The main benefit of having overlapping windows is to have smoother transitions from one observation to the next; the challenge, however, is then deciding how to make the labels for the supervised learning models. If there is an anomaly in half of one window and no anomaly in the other half of the window, should we consider that observation anomalous

or normal? We also plan to investigate the requirement that windows must all be of the same length. That is, we plan to explore adaptive windowing methods that extract windows of different lengths depending on the underlying signal. The goal with adaptive windowing is to create observations that correspond to events rather than a simple time interval.

## 6.2 Number and Placement of Sensors

This work, along with [1–3, 25], has shown that machine learning algorithms can successfully separate normal and anomalous observations in passive seismic data from geophones. In Chapter 3, we showed that an anomaly detection system relying on the passive seismic data from a single geophone can vary widely in performance depending on the choice of the sensor in the structure. Then, in Chapter 4, we showed that combining the spectral features extracted from each sensor into the same observation yields models that perform reliably well. We have not, however, shown mathematical analysis of the minimum number of sensors actually needed to perform well on the learning task. Nor have we tried to identify the best way to arrange the grid of sensors (i.e., how far apart should the sensors be?).

To answer these questions in ways that generalize beyond the small-scale experimental data set used in our study, we plan to gather and study more data sets from large-scale embankments. The goal of this work is to investigate how the size of the embankment impacts the necessary number of sensors and optimal placement pattern of those sensors.

## 6.3 Other Data Sets

We also propose to apply our technique of using data from more than one geophone sensor to other experimental passive seismic data sets for anomaly detection in EDLs. For instance, the IJkdijk full-scale test embankment in the Netherlands was given reservoir loads known to induce internal erosion events in real world EDLs [41]. The labels for the IJkdijk data set are not as high quality as the labels available in the USBR piping data set. Since the IJkdijk experiment was at a much larger (full) scale than the laboratory piping experiment, the IJkdijk data set is likely to contain much more noise than the laboratory data. We propose

to investigate whether our technique presented in this thesis is able to predict anomalies in the IJkdijk data set.

We could also utilize data from a crackbox test where a vertical array of 12 geophones was placed in an experimental embankment. A hydraulic lift underneath the test embankment increased pressure in small increments to induce cracking events [3]. Additionally, there is passive seismic data available from the full scale Colijnsplat levee in the Netherlands, which contains seepage and other erosion events [3]. Applying our anomaly detection techniques to these data sets would allow investigation of (1) whether the techniques work at full-scale and (2) our ability to detect different types of erosion under a wider variety of environmental conditions.

## REFERENCES CITED

- [1] Wendy Fisher, Tracy Camp, and Valeria Krzhizhanovskaya. Detecting erosion events in earth dam and levee passive seismic data with clustering. In *2015 IEEE 14th International Conference on Machine Learning and Applications*, pages 903–910. IEEE, 2015. URL <http://dx.doi.org/10.1109/ICMLA.2015.9>.
- [2] Wendy Fisher, Tracy Camp, and Valeria Krzhizhanovskaya. Anomaly detection in earth dam and levee passive seismic data using support vector machines and automatic feature selection. *Journal of Computational Science*, 20:143–153, 2017. URL <https://www.sciencedirect.com/science/article/pii/S1877750316304185>.
- [3] Wendy Fisher, Tracy Camp, and Valeria Krzhizhanovskaya. Crack detection in earth dam and levee passive seismic data using support vector machines. In *2016 International Conference on Computational Science*, volume 80, pages 577–586, 2016.
- [4] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [5] Fatalities as Northeast Brazilian Dam Bursts, 2009. URL <https://www.internationalrivers.org/blogs/232/fatalities-as-northeast-brazilian-dam-bursts>. (Accessed January 30, 2017).
- [6] Laura Ruhl, Avner Vengosh, Gary Dwyer, Heileen Hsu-Kim, Amrika Deonarine, Mike Bergin, and Julia Kravchenko. Survey of the potential environmental and health impacts in the immediate aftermath of the coal ash spill in Kingston, Tennessee. *Environmental Science Technology*, (43):6326–6333, 2009.
- [7] AECOM. Root Cause Analysis of TVA Kingston dredge pond failure on December 22, 2008, June 2009. URL [https://web.archive.org/web/20100528045328/http://www.tva.gov/kingston/rca/FINAL%20REPORT-062409/vol1/Vol\\_1-Text/FINAL-062509-Failure\\_Analysis\\_Report\\_VolI.pdf](https://web.archive.org/web/20100528045328/http://www.tva.gov/kingston/rca/FINAL%20REPORT-062409/vol1/Vol_1-Text/FINAL-062509-Failure_Analysis_Report_VolI.pdf). (Accessed October 14, 2017).
- [8] Tennessee Valley Authority Kingston Fossil Plant coal ash release natural resource damage assessment: Restoration and compensation determination plan. <https://www.fws.gov/cookeville>, 2016. (Accessed February 25, 2017).
- [9] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1997.
- [10] Adithya Uligere Narasimhamurthy. *Performance Analysis of Wireless Sensor Networks in Geophysical Sensing Applications*. Master’s thesis, Colorado School of Mines, 2016.

- [11] Joel Grus. *Data Science from Scratch: First Principles with Python*. O’Reilly, Beijing, 2015.
- [12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, July 2009. URL <http://doi.acm.org/10.1145/1541880.1541882>.
- [13] Giovanni Seni and John Elder. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. Morgan and Claypool Publishers, 2010.
- [14] Andreas Müller and Sarah Guido. *Introduction to Machine Learning with Python : A Guide for Data Scientists*. O’Reilly, October 2016.
- [15] Jerome H Friedman, Bogdan E Popescu, et al. Importance sampled learning ensembles. *Journal of Machine Learning Research*, 94305, 2003.
- [16] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. URL <http://dx.doi.org/10.1007/BF00058655>.
- [17] Yoav Freund and Robert E. Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, pages 23–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. URL [http://dx.doi.org/10.1007/3-540-59119-2\\_166](http://dx.doi.org/10.1007/3-540-59119-2_166).
- [18] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. URL <http://dx.doi.org/10.1023/A:1010933404324>.
- [19] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [20] Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99 – 111, 2005. URL <http://www.sciencedirect.com/science/article/pii/S156625350400034X>.
- [21] Tetrattech report: City of Roseville Multi-Hazard Mitigation Plan, 2011. <http://www.roseville.ca.us/civicax/filebank/blobdload.aspx?blobid=19067> (Accessed on January 30, 2017).
- [22] G. Jan Schiereck. Fundamentals on water defences. Technical report, TAW-ENW, 1998. URL <https://repository.tudelft.nl/islandora/object/uuid:fe16f99c-6ddc-49bd-b7f7-18223e9b73b4/?collection=research>. (Accessed on January 28, 2017).

- [23] Mark Foster, Robin Fell, and Matt Spannagle. The statistics of embankment dam failures and accidents. *Canadian Geotechnical Journal*, 37(5):1000–1024, 2000. URL <http://www.nrcresearchpress.com/doi/abs/10.1139/t00-030>.
- [24] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *Neural Information Processing Systems*, volume 12, pages 582–588, 1999.
- [25] Wendy Fisher, Blake Jackson, Tracy Camp, and Valeria Krzhizhanovskaya. Multivariate Gaussian anomaly detection in earth dam and levee passive seismic data. To appear in *2017 IEEE 16th International Conference on Machine Learning and Applications*, IEEE, 2017.
- [26] UrbanFlood: About UrbanFlood. <http://www.urbanflood.eu/Pages/aboutus.aspx>, 2012. (Accessed February 1, 2017).
- [27] Alexander L. Pyayt, Alexey P. Kozionov, Ilya I. Mokhov, Bernhard Lang, Robert J. Meijer, Valeria Krzhizhanovskaya, and Peter M. A. Sloom. Time-frequency methods for structural health monitoring. *Sensors*, 14(3):5147–5173, 2014. URL <http://www.mdpi.com/1424-8220/14/3/5147>.
- [28] Lalitha Dabhiru, James V. Aanstoos, Majid Mahrooghy, Wei Li, Arjun Shanker, and Nicolas H. Younan. Levee anomaly detection using polarimetric synthetic aperture radar data. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 5113–5116, July 2012.
- [29] California Institute of Technology Jet Propulsion Laboratory. Flight planning help. <https://uavsar.jpl.nasa.gov/help/faq.html>, 2013 (Accessed February 25, 2017).
- [30] M.J. Rubin. *Efficient and Automatic Wireless Geohazard Monitoring*. PhD thesis, Colorado School of Mines, 2014.
- [31] Kerri Stone, Charles Oden, Brian Hoenes, and Tracy Camp. Hardware for a wireless geophysical monitoring testbed. In *2011 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 652–659, 2011.
- [32] Olivier Lartillot and Petri Toiviainen. A Matlab toolbox for musical feature extraction from audio. In *International Conference on Digital Audio Effects*, pages 237–244, 2007.
- [33] Oliver Lartillot. *MIR Toolbox 1.6.1 : User’s Manual*. Aalborg University, December 2014.

- [34] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>, 2003. (Accessed February 25, 2017).
- [35] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine learning*, 36(1):85–103, 1999.
- [36] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- [37] Gilles Louppe and Pierre Geurts. Ensembles on random patches. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 346–361. Springer, 2012.
- [38] S. Duvvuri and B. Singhal. *Spark for Data Science*. Packt Publishing, 2016. URL <https://books.google.com/books?id=hIRcDgAAQBAJ>.
- [39] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.
- [40] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240. ACM, 2006. URL <http://doi.acm.org/10.1145/1143844.1143874>.
- [41] Michael A Mooney, Minal L Parekh, Ben Lowry, Justin Rittgers, Jacob Grasmick, André R Koelewijn, André Revil, and Wei Zhou. Design and implementation of geophysical monitoring and remote sensing during a full-scale embankment internal erosion test. In *Geo-Congress 2014: Geo-characterization and Modeling for Sustainability*, pages 202–211, 2014. URL <http://ascelibrary.org/doi/abs/10.1061/9780784413272.021>.