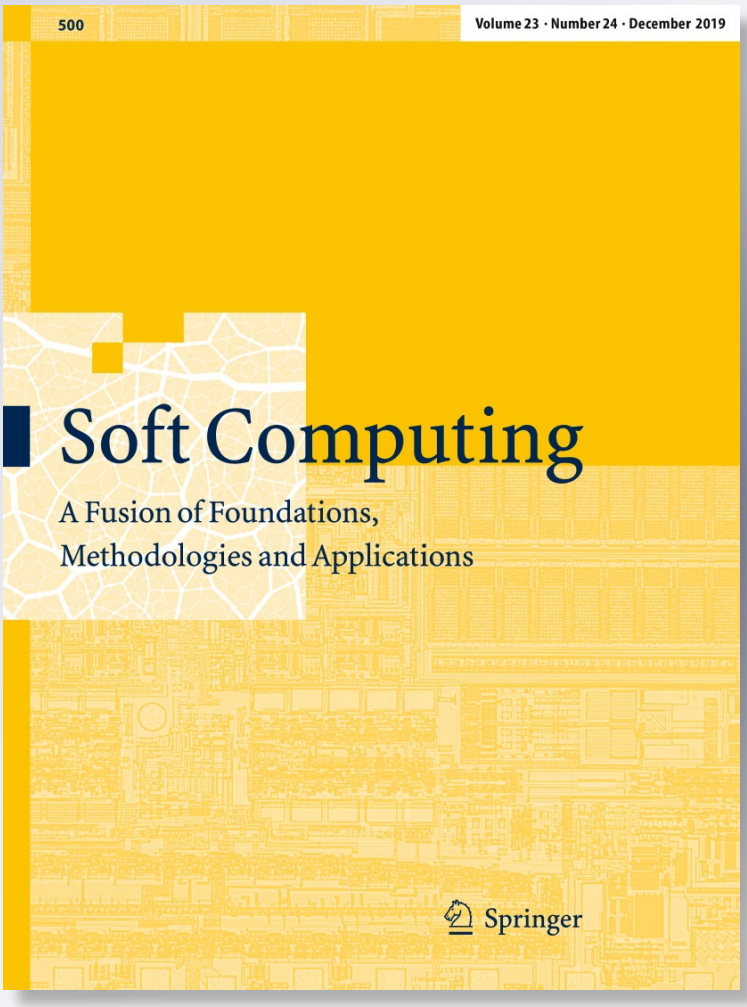
Analysis of remote sensing imagery for disaster assessment using deep learning: a case study of flooding event

Liping Yang & Guido Cervone

Soft Computing
A Fusion of Foundations,
Methodologies and Applications

ISSN 1432-7643
Volume 23
Number 24

Soft Comput (2019) 23:13393-13408
DOI 10.1007/s00500-019-03878-8



Springer

Soft Computing (2019) 23:13393–13408
https://doi.org/10.1007/s00500-019-03878-8

METHODOLOGIES AND APPLICATION

Analysis of remote sensing imagery for disaster assessment using deep learning: a case study of flooding event

Liping Yang[1] · Guido Cervone[1]

Published online: 7 March 2019

Abstract
This paper proposes a methodology that integrates deep learning and machine learning for automatically assessing damage with limited human input in hundreds of thousands of aerial images. The goal is to develop a system that can help automatically identifying damaged areas in massive amount of data. The main difficulty consists in damaged infrastructure looking very different from when undamaged, likely resulting in an incorrect classification because of their different appearance, and the fact that deep learning and machine learning training sets normally only include undamaged infrastructures. In the proposed method, a deep learning algorithm is firstly used to automatically extract the presence of critical infrastructure from imagery, such as bridges, roads, or houses. However, because damaged infrastructure looks very different from when undamaged, the set of features identified can contain errors. A small portion of the images are then manually labeled if they include damaged areas, or not. Multiple machine learning algorithms are used to learn attribute–value relationships on the labeled data to capture the characteristic features associated with damaged areas. Finally, the trained classifiers are combined to construct an ensemble max-voting classifier. The selected max-voting model is then applied to the remaining unlabeled data to automatically identify images including damaged infrastructure. Evaluation results (85.6% accuracy and 89.09% F1 score) demonstrated the effectiveness of combining deep learning and an ensemble max-voting classifier of multiple machine learning models to analyze aerial images for damage assessment.

Keywords Spatiotemporal data · Image classification · TensorFlow · Machine learning · Deep learning · Damage assessment

Abbreviations

ML	Machine learning
DL	Deep learning
CAP	Civilian Air Patrol
AI	Artificial intelligence
CNN	Convolutional neural network
RNN	Recurrent neural network
MLP	Multilayer perceptron
SVM	Support vector machine
RBF	Radial basis function
DT	Decision tree
NB	Naive Bayes
KNN	k-Nearest neighbors
RF	Random forest
GB	Gradient boosting
GBC	Gradient boosting classifier
LR	Logistic regression
LDA	Linear discriminant analysis
NN	Neural networks
USGS	United States Geological Survey
USGS HDDS	USGS Hazards Data Distribution System

Communicated by V. Loia.

✉ Liping Yang
liping.yang@psu.edu

Guido Cervone
cervone@psu.edu

[1] Department of Geography and Institute for CyberScience, The Pennsylvania State University, University Park, PA 16802, USA

1 Introduction

During disasters, large volumes of images are routinely captured by officials and often freely contributed by citizens. The availability of high-resolution satellite and aerial platforms, paired with the recent increase in unmanned aerial systems and mobile devices, has led to the generation of massive, distributed, spatiotemporal data. Our ability to generate data

greatly exceeds our ability to analyzing them, leading to a data-rich but knowledge-poor environment.

One of the main difficulties during disasters is the need to obtain actionable knowledge from potentially massive and unstructured data. In particular, large collection of imagery presents several advantages and challenges. On the one hand, they provide a broad view of a disaster area, which can potentially provide mission critical knowledge. On the other hand, however, they present several challenges because images are collected using different sensors, angles, illumination, and weather conditions, which make the automatic extraction of knowledge a challenging task. Furthermore, disaster areas present unique challenges because damaged infrastructure looks very different from their undamaged appearance. Very few examples exist of damaged infrastructure, as the data are only now being collected for the first time.

For applications based on supervised machine learning (ML), including deep learning (DL), the production of labeled data is often a bottleneck because it requires extensive human intervention. The process of generating training examples is time-consuming, prone to errors, and a major obstacle to the full automation of tasks. Furthermore, for some tasks, labeling data requires highly skilled domain experts (Weiss and Provost 2003; Xiao et al. 2015; Zhu et al. 2017). The need for large labeled data is particularly stressed in deep learning, where the advantage of being able to automatically learn very complex concepts comes with the requirement of massive amount of manually annotated examples and counter examples.

The main goal of this research is to understand the 'mistakes' that a deep learning algorithm makes when employed to classify images showing damaged areas, and use these mistakes to better predict the areas with most damage. Therefore, what was referred as a 'mistake' is just a misclassification of the deep learning algorithm that is due to the absence of damaged areas in the training set. The main rationale is that mistakes are useful when they are predictable.

In this article, we have developed a method that uses DL for the automatic extraction of features from remote sensing images, and multiple ML classifiers that use these features along with geographical features that come with the imagery data to automatically predict areas of damage from unseen image data. To accomplish this task, a small set of images were manually labeled as showing a flooded area, or not. The most characteristics features associated with the flooded class were identified using several ML classifiers, and an ensemble max-voting model that combines multiple ML classifier was then used to classify the unlabeled images. In this article, image classification does not refer to assigning each individual pixel to a class (e.g., vegetation, water), but rather to assign the entire image to a specific class (e.g., flooded vs. not flooded).

The proposed methodology was applied to the automatic identification of flooded areas using 22,891 aerial images collected by the Civilian Air Patrol (CAP) during the 2015 floods that affected the state of Texas.

The remainder of this article is organized as follows. Section 2 highlights related work from both ML and DL perspective and flooding damage assessment aspects. Section 3 describes the proposed methodology, including evaluation techniques and metrics. To demonstrate our methodology using real-world data set, a case study along its results is presented in Sect. 4. The paper concludes in Sect. 5 with discussion of application potential of the proposed methodology.

2 Related work

In this section, we review some of the previous work that bears on our contribution, from ML and DL perspectives and from damage assessment (particularly flooding events) aspects.

Machine learning (ML) is a branch of artificial intelligence (AI). ML focuses on algorithms that can learn how to perform a task from and make predictions on data without explicitly programmed instructions as conventional programming does. However, ML relies on *feature engineering*, which is the process of using domain-specific knowledge to manually create features from data (Domingos 2012; Yang et al. 2018). The extracted features, along with a set of labeled data that have those features, are then used to train a ML algorithm. The trained model can be used to make predictions for new unseen data. The process of feature engineering is often difficult and time-consuming.

Deep learning (DL) is a branch of ML. DL can automatically learn task-relevant features from a big set of labeled data. Thus, DL bypasses the tedious process of feature engineering that is essential for ML, but it requires much larger labeled data sets compared with ML (Yang et al. 2018). The big volume of annotated data required for training a DL model with good performance is the main bottleneck for DL (Yang et al. 2018). This is one of the main reasons that motivate us to develop the method we propose in this article (see Sect. 3) to produce good performance ML/DL models with a reduced amount of labeled data. Recent rapid advances in ML and especially in DL, plus the availability of many open sourced resources (e.g., Google TensorFlow (Abadi et al. 2016), an open-source software library for machine learning and deep learning), provide the potential to leverage spatiotemporal big data to advance geospatial research and solve relevant application challenges.

Different DL algorithms have been developed, among which the *convolutional neural network (CNN)* (LeCun et al. 1998) and the *recurrent neural network (RNN)* (Elman 1990)

and their variants are most successfully and commonly used. A general conclusion in the literature (Krizhevsky et al. 2012; Simonyan and Zisserman 2014; LeCun et al. 2015; Szegedy et al. 2016; Gu et al. 2017) about deep learning algorithms is that CNNs are good at handling image and video data and that RNNs shine for sequence data such as text. CNNs integrate automatic feature extraction and discriminative classifier in one model. A major reason that leads to the increasing success for image recognition and analysis using DL is the availability of big image repositories, such as ImageNet (Deng et al. 2009), that support such work for benchmarking. ImageNet (Deng et al. 2009) is an image database that contains 3.2 million labeled images and spreads over 1000 categories. For example, well-known pre-trained CNN models *AlexNet* (Krizhevsky et al. 2012), *VGGNet* (Simonyan and Zisserman 2014), *GoogLeNet* (Szegedy et al. 2015), and *Inception-v3* (Szegedy et al. 2016) are all trained on the ImageNet, among which Inception-v3 achieved the best performance (Szegedy et al. 2016).

Many ML classifiers have been developed in the literature. (ML algorithms implemented for classification tasks are called classifiers.) Witten et al. (2011) introduced basic ML and data mining methods, which include decision trees (DTs), support vector machines (SVMs), and naive Bayes (NB) classifiers; Amancio et al. (2014) carried out a systematic comparison of nine well-known ML supervised classifiers, including multilayer perceptron (MLP), Bayesian network, SVM, NB, k-nearest neighbors (KNN). Below, we briefly review the ML classifiers that we use in the case study (elaborated in Sect. 4).

K-nearest neighbors (KNN) algorithm is the simplest among all ML algorithms (Domingos 2015). KNN is a nonparametric method used for classification and regression (Altman 1992). It is a type of instance-based learning (also called 'lazy learning'), where the function is only approximated locally and generalization of the training data is delayed until classification (Daelemans and Van den Bosch 2005). Being a nonparametric method, KNN is often successful in classification situations where the decision boundary is very irregular (Domingos 2015). In low feature dimensions (e.g., two or three), KNN usually works quite well. But as the number of dimensions goes up, things fall apart quickly (Domingos 2015). Thus, KNN is the most useful for large data sets with a small number of features.

Decision trees (DTs) are a nonparametric supervised learning method that can be applied to both classification and regression problems (Quinlan 1986; Han et al. 2011; Bishop 2006). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Advantages of DTs are as follows: They can handle both numerical and categorical data; they can deal with information that is noisy and/or incomplete (Quinlan 1986). Limitations of DTs are as follows: They can create complex trees that do not generalize the data well (i.e., overfitting) (Domingos 2012); They would create biased trees if some classes dominate. A multi-disciplinary survey about automatic construction of DTs from data was investigated in Murthy (1998), where important similar issues and heuristics for DT construction from different problems are identified and discussed. A framework for sensitivity analysis of DTs can be found in Kamiński et al. (2018).

Ensemble methods average the predictions of multiple ML algorithms to get a better prediction than those obtained from any of the algorithms alone (Opitz and Maclin 1999; Polikar 2006; Rokach 2010). Two main techniques for ensemble learning methods are *bagging* (i.e., **b**ootstrap **agg**regat**ing**) (Breiman 1996) and *boosting* (Freund and Schapire 1997). Bagging uses bootstrapping, through training several different models independently, each using a different set of samples and then averaging their predictions (Breiman 1996). In bagging methods, the averaged model is in general better than any of the single model as its variance is decreased. By contrast, boosting involves incrementally generating an ensemble by training each model to emphasize the training samples that have been classified incorrectly in previous models, through reweighing those samples with much higher weight (Gislason et al. 2006). In boosting methods, multiple models are trained sequentially and the goal is to reduce the bias of the averaged model.

Random forests (RFs) are an ensemble learning technique that builds multiple trees based on random bootstrapped samples of the training data (Breiman 2001). RF combines random DTs with bagging to achieve high classification accuracy (Breiman 1996). RFs in general do not suffer from overfitting (as encountered by DTs), as the resampling is not based on weighting (Gislason et al. 2006); RFs are not sensitive to noise (Gislason et al. 2006) and do not require long training time (Breiman 2001), because RFs train each tree independently, using a random sample from the training data, and thus are good for parallelism or distributed computing. *Gradient boosting (GB)* is a ML algorithm that combines gradient descent and boosting (Friedman 2001; Hastie et al. 2009); it is an ensemble learning method for improving the predictive performance of classification or regression procedures, such as DTs (Hastie et al. 2009). The advantages of GB are as follows: (1) predictive power and (2) robustness to outliers in output space (Hastie et al. 2009). Limitations of GB: scalability; due to the sequential nature of boosting, it is difficult to be parallelized (Hastie et al. 2009). Thus, for data sets with a large number of classes, RF classifier as an alternative to GB classifier is recommended.

Naive Bayes (NB) is a simple but also fast and reliable probabilistic algorithm using Bayes' theorem (Domingos and Pazzani 1997). NB algorithms assume that the features relevant to the annotated samples are independent (Domingos and Pazzani 1997; Hastie et al. 2009). NB classifiers are scal-

able due to their simplicity (Russell et al. 2003). *Logistic regression (LR)* is a linear algorithm for classification task rather than regression (Bishop 2006). LR algorithm estimates the probability that describes the possible outcomes of a binary classification task based on input features using a logistic function. Specifically, LR, analogous to NB, extracts a set of weighted features from the input annotated samples, takes logs, and then combines them linearly. LR is robust against extreme data points (e.g., outliers). The most important difference between NB and LR is that LR is a discriminative classifier, while NB is generative (Ng and Jordan 2002). LR works well when the classes in given training data set are linearly separable; however, in general, it tends not to perform well when there are multiple or nonlinear decision boundaries.

Linear discriminant analysis (LDA) is a classification algorithm. The goal of LDA is to reduce dimensionality of input samples and meanwhile maximize the discrimination information among classes through generating discriminant function (Bishop 2006; Hastie et al. 2009). LDA assumes that the same covariance matrix is shared across classes, but LR does not require such assumption; thus, LR is a more robust algorithm than LDA (Press and Wilson 1978; Liong and Foo 2013). However, Hastie et al. pointed out in Hastie et al. (2009) that LDA and LR generate similar classification results, even when LDA is used inappropriately (e.g., assumptions are not met).

Support vector machines (SVMs) (Hastie et al. 2009) are supervised learning algorithms used for both classification and regression analysis. SVMs can find global optimum, because finding global optimum is a convex optimization problem for SVMs. Thus, it can result in a unique solution. This is an advantage compared with neural networks (NN, elaborated below), which have multiple solutions associated with local minima (Shawe-Taylor and Cristianini 2004). One strength of SVM is that SVM algorithms are also fairly robust against overfitting, especially in high-dimensional feature space. Another key advantage of SVMs is the use of kernel functions to solve nonlinearly separable data set (Shawe-Taylor and Cristianini 2004). Thus, one strength of SVMs typically comes from using nonlinear kernels to model nonlinear decision boundaries. However, the biggest limitation of SVMs lies in choice of the right kernel function for a given problem (Burges 1998), even though in the literature, radial basis function (RBF) kernel (Buhmann 2003) is the most commonly used kernel function. In addition, SVMs do not scale well to large data sets.

A *neural network (NN)* is a set of connected input/output (neural) nodes with weighted connections between the nodes (Bishop 2006; Hastie et al. 2009; Han et al. 2011). It can be used for both classification and regression problems. For classification tasks, during the learning phase, the NN learns by adjusting the weights according to given labeled data samples, in order to predict the correct class label of the unseen data sample. When the number of hidden layers of a NN is ≥ 2, it is called a deep NN (i.e., deep learning). NNs require long training time and large set of training samples to get a trained model with good performance. However, NN algorithms are inherently parallel, so parallelization techniques can be used to speed up its computational process (Han et al. 2011).

Wainer (2016) compared 14 different classification algorithms (including RF, GBC, linear SVM, SVM with RBF kernel, 1-hidden-layer NN, KNN, NB, LDA, and LR) on 115 real-world binary datasets. (Binary here refers to binary classification; for example, the label of flooding prediction is flooded or not.) Wainer (2016) concluded that RF, SVM with RBF kernel, and GBC are the top three ML classifiers that most likely result in the highest accuracy, especially between RF and SVM with RBF kernel. In terms of training and testing execution times, SVM with a RBF kernel is faster than RF and GBC.

On damage assessment aspect, although there is now a substantial body of work, there is somewhat less on using DL for damage assessment for flood events and how the analysis result might be generated automatically. Most existing methods are based on social media data such as Twitter and Flickr, and these are now reviewed.

Cervone et al. (2016) proposed a method that leverage social media such as Flickr and Twitter for tasking the collection of remote sensing imagery during disasters for damage assessment. Panteras and Cervone (2018) proposed a methodology to compensate for missing satellite data using social media for the 2015 flooding in North Carolina. Li et al. (2018) also used social media for the data source for flood mapping, where a kernel-based flood mapping model was developed to map the flooding possibility for their study area based on the water height points derived from tweets and stream gauges. The identified patterns of Twitter activity were used to assign the weights of flood model parameters. However, social media data source is not reliable, simply because it cannot be guaranteed the volume of and geolocation availability of social media data.

3 Methodology

This article proposes a methodology that leverages the power of DL for automatic feature extraction and does not require a large set of labeled data to produce a ML model with good performance for a specific domain task (e.g., flooded image classification). Figure 1 shows an overview of the method, including the main components and flows between them.

Our method starts with a big set of unlabeled images, which then are fed into a DL model pre-trained on ImageNet (Deng et al. 2009) using TensorFlow to identify features.

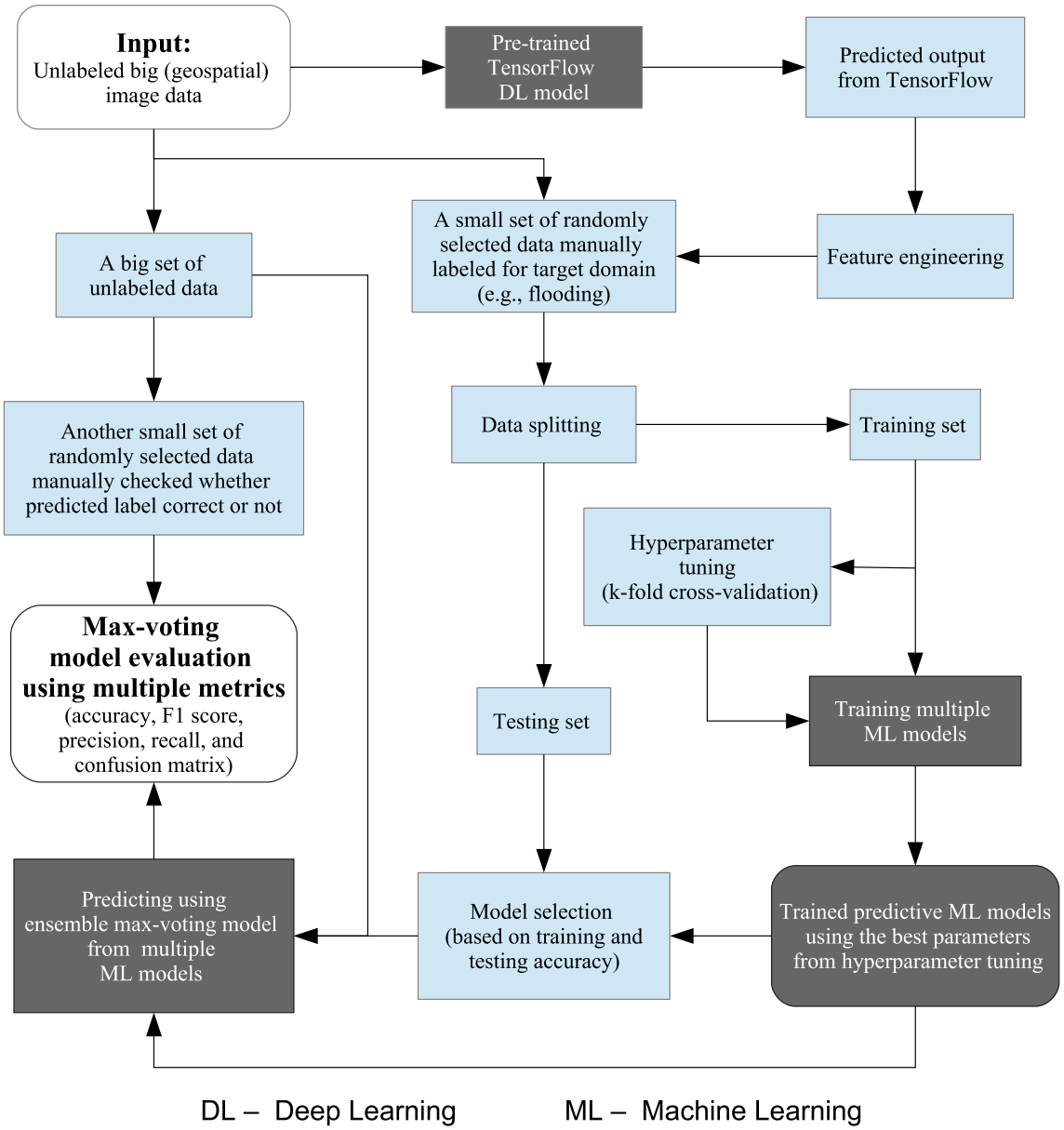


Fig. 1 Flowchart of our methodology (see Sect. 2 for feature engineering)

Specifically, we use a pre-trained DL CNN model, Inception-v3 (Szegedy et al. 2016), which has been trained on the ImageNet data set (Deng et al. 2009) by Google using TensorFlow. The pre-trained Inception-v3 model can differentiate among 1000 different classes from images, such as "bridge", "boathouse", or "plane". We selected the model because Inception-v3 achieved best performance on ImageNet data set (see Sect. 2).

We then manually label a small set of randomly selected images from the big unlabeled data set. Once the features are identified by the DL model, we use them in combination with the small set of manually labeled data to train multiple ML algorithms. We select the top 20 TensorFlow output, plus, latitude and longitude, as features (see Sect. 2 for details about feature engineering). Adding specific geographical variables (longitude and latitude) as predictors can have some strong implications toward the classification output. They have the limiting effect of making the classification learned applicable only to a certain geographical area. If data for new areas need to be classified, it is likely that an entire new classification task must be performed. On the other hand, the geographical proximity to flooded and not-flooded areas is likely a very discriminant predictor that can drastically increase the accuracy. In an operational setting, it is usually known areas that are affected and those that are not, and because our methodology is based, partially, on manual labeling, it is safe to

assume that geographical information is known. This is not a limiting factor in the methodology, as the geographical variables can be omitted if unavailable, or if a general model is required.

The next step is to split the small manually annotated data set into two subsets (e.g., 70% for training, 30% for testing). It is useful due to its speed, simplicity, and flexibility. Generally, there is not a universally better ML algorithm for all problems (Wolpert 1996; Salzberg 1997; Caruana and Niculescu-Mizil 2006; Domingos 2015). The main reason is that learning from finite labeled samples requires making assumptions, and different ML algorithms make different assumptions (a.k.a. bias) (Domingos 2012, 2015). It is necessary to test which algorithm works best with a specific problem. Thus, we train the images in the training set (with the selected features) on nine well-known ML algorithms for classification tasks (specifically KNN, GBC, RF, DT, NB, LDA, LR, SVM, and NN).

To make each ML model perform their 'best,' before training the ML algorithms, we tune hyperparameters of each ML algorithm on the training set (see details in Sect. 4.5). We use k-fold cross-validation (Dubitzky et al. 2007; Bird et al. 2009; Hastie et al. 2009; Witten et al. 2011) while fine-tuning the hyperparameters to avoid overfitting, because it allows performance variation across training sets to be examined and thus can test how well the trained ML algorithms is able to handle larger unseen data samples. In addition, k-fold cross-validation can make good use of the small set of labeled data, because all the labeled data are used for training and testing in different folds through systematically creating multiple train/test splits and averaging the results.

After fine-tuning, we use the best hyperparameters of each ML algorithm to train on the training set and the models are evaluated on the testing set. The top performance ML model is selected as the optimal model to predict all the rest of unlabeled images.

The fundamental goal of ML is to generalize or abstract beyond the examples in the training set (Domingos 2012). The primary objective of model evaluation is to estimate how well a model will perform on unseen data (i.e., those data samples not in the training set). After performing model comparison, we select the top performance ensemble max-voting model that combines multiple ML algorithms for prediction. The selected model is further evaluated using another small set of manually checked images randomly selected from the unlabeled set of images with multiple evaluation metrics (elaborated below), including accuracy, F1 score, precision, recall, and confusion matrix.

Bird et al. (2009) and Sokolova and Lapalme (2009) provide a comprehensive introduction to different evaluation metrics for classification tasks. For classification problems, accuracy is the most commonly used and the simplest model evaluation metric; it measures the percentage of inputs in the test set that are correctly classified by the classifier (Bird et al. 2009; Han et al. 2011). Thus, the evaluation metric we used in this article is accuracy while performing model comparison. A confusion matrix (Provost and Kohavi 1998) summarizes the performance of a classification algorithm. It contains information about actual and predicted classifications and helps to determine what the classification model is getting correct and what types of errors it is making. (Thus, it provides insight not only into the errors being made by the classifier but more importantly the types of errors that are being made.) In addition, standard machine learning evaluation metrics include precision (TP/(TP + FP)), recall (TP/(TP+FN)), and F1 score (a weighted average of the precision and recall—a number between 0 and 1 that explains how well the network performed where reaching its best value at 1 and the worst at 0), where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

4 Case study: aerial image classification for flooding hazard

The proposed method was used to classify thousands of aerial images relative to a flood event.

4.1 Data

The data used in this research are a collection of aerial imagery collected from airplanes (CAP images) relative to the 2015 floods in Texas. The imagery is available through the USGS Hazards Data Distribution System (HDDS) Explorer[1] Website. The data comprise of 22,891 CAP images, collected between May 16, 2015, and June 23, 2015.

4.2 Feature extraction using TensorFlow DL model

We used Google Inception-v3 pre-trained DL model on ImageNet to identify and extract features using TensorFlow from the whole set of data described in Sect. 4.1. We chose top 20 TensorFlow outputs plus latitude and longitude as the set of features for training ML models. Features detected by TensorFlow DL model can tell us what appear in the aerial images, and geospatial location information (e.g., latitude and longitude) bears important clue for flooding classification. Figure 2 presents examples of the top two TensorFlow outputs using the TensorFlow pre-trained DL model. Those are two of the 22 features we used to train different ML models.

[1] USGS Hazards Data Distribution System (HDDS) Explorer https://hddsexplorer.usgs.gov/.

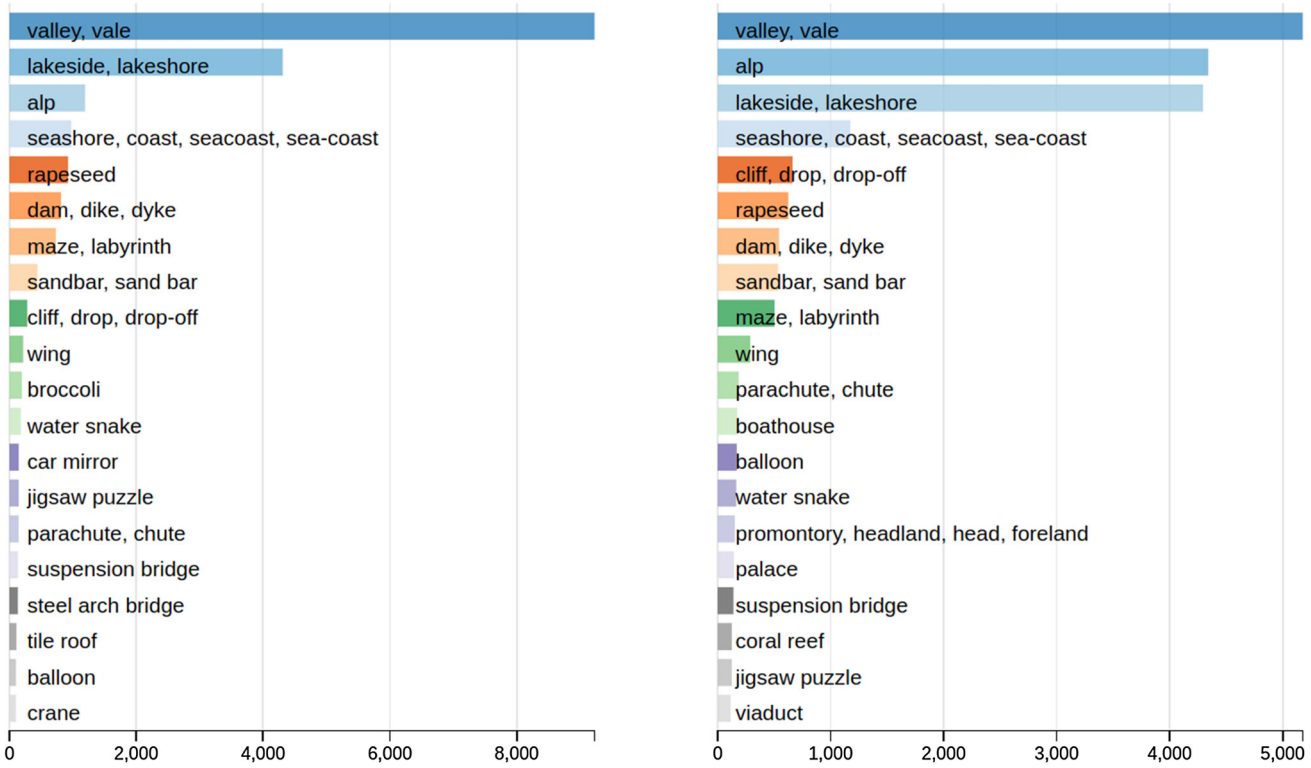


Fig. 2 Row charts for top two TensorFlow output features for all aerial images (left: first; right: second) using the pre-trained Inception-v3 DL model. Top 20 TensorFlow output features, plus geospatial information (i.e., latitude and longitude), are the feature set we used to train a series of ML models

4.3 Manual annotation of aerial images as flooded or not

In order to train ML models, besides the 22 selected features described in Sect. 4.2, it is necessary to manually label a small set of aerial images as either *flooded* or *not flooded*. So the ML learners can learn from the small set of labeled images along with the 22 features and produce a model with good predictive performance.

We have developed an interactive Web app for manual annotation of aerial images. We randomly selected 1000 images for manual annotation of whether an aerial image is flooded. Among the 1000 labeled images, 613 images are flooded and 387 not flooded. Figure 3 presents the visual geospatial distribution of the randomly selected images that are manually labeled. See Fig. 4 for examples of manually labeled flooding images with its top 3 TensorFlow outputs. (Annotators do not see the TensorFlow outputs while they manually label the images.)

4.4 Encoding categorical features

As introduced in Sect. 4.2, we have 22 selected features, but among which the type of the 20 features from TensorFlow outputs is categorical (as shown in Fig. 2). Categorical features are one common type of non-numerical data. This type of features cannot be directly fed into ML models, because all ML models require numerical data.

To encode the 20 categorical features (i.e., convert the categorical values to numerical values), we first get all categorical values (the 1000 class names from ImageNet dataset) and map those class labels to integer numbers (in our case, mapping 1000 class names to integer 0 and 999). Then, we use the mapping to encode all categorical features (including those features extracted from the images in the training set and also in the testing set, as well as in the larger set of unlabeled images). It does not matter which class label is mapped to which integer number, but it is important to keep the mapping consistent; that is, make sure to use the same mapping to encode all of the involved data for all relevant tasks, in order to make the performance of different ML models meaningful and comparable.

4.5 Hyperparameter tuning and performance comparison of multiple ML algorithms using optimized hyperparameters

Hyperparameter tuning is the process of finding the set of hyperparameter values of a machine learning algorithm that produces the best model results. Hyperparameter tuning is

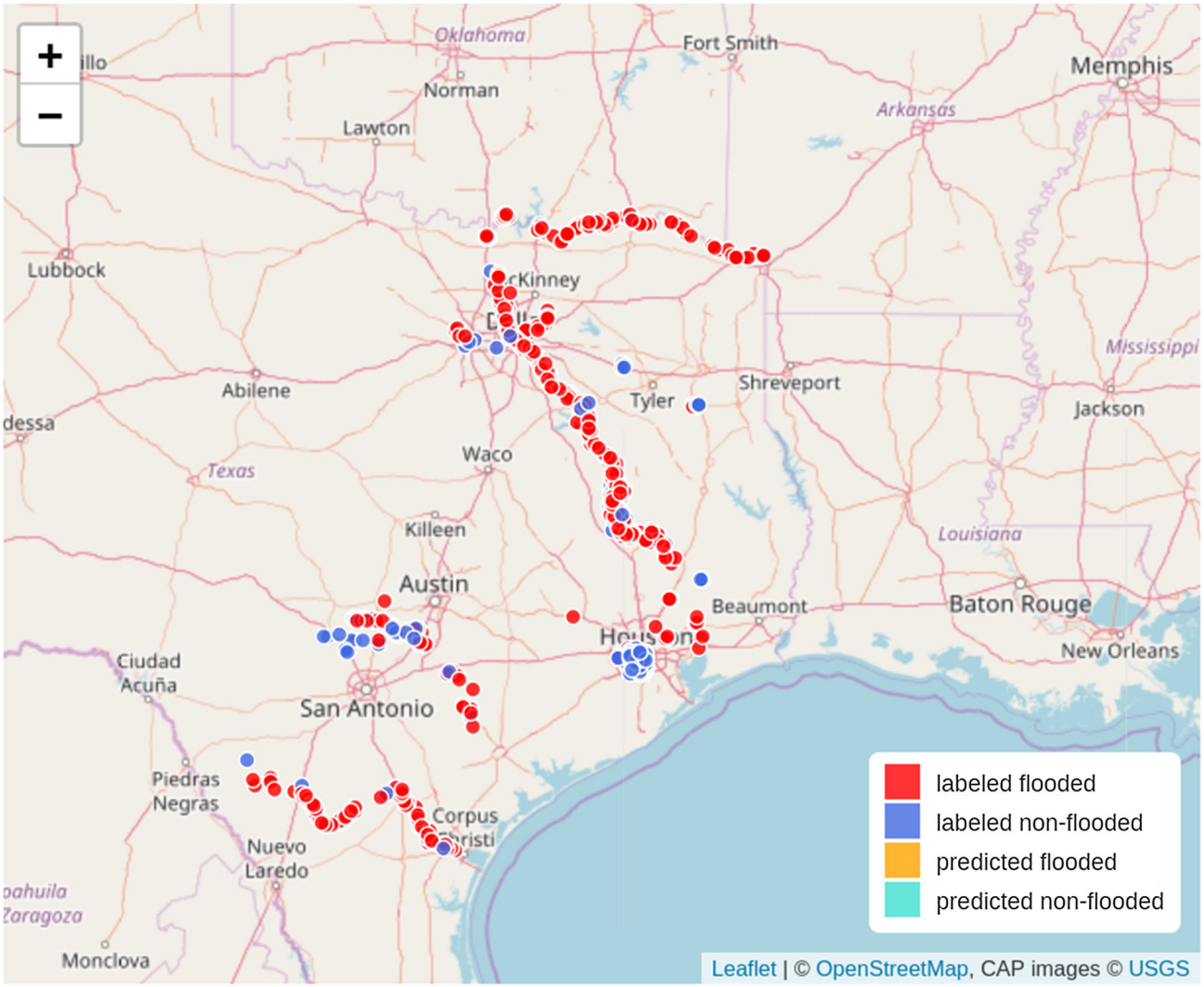


Fig. 3 The geospatial distribution of the 1000 randomly selected labeled images

crucial because it is used to search for the best hyperparameters of a machine learning algorithm for a given dataset.

In this research, we use grid search provided in Scikit-Learn (Pedregosa et al. 2011) to tune hyperparameters of the ML algorithms we used. Detailed settings about the hyperparameter tuning for each ML algorithm are provided in Appendix A, where we also gave tuning results including best hyperparameters and performance comparison with baseline model that using the default hyperparameters.

As we introduced in Sect. 3, there is no single ML algorithm that would work best for all problems. Thus, we trained nine commonly used ML algorithms (GBC, RF, DT, NB, LDA, LR, SVM, KNN, and NN) on our case study data set. (Those algorithms are elaborated in Sect. 2.) For NB, specially, we used Gaussian NB. For SVM, in particular, we used SVC (Ben-Hur et al. 2001), and the default kernel is the (Gaussian) radial basis function (RBF) kernel (Buhmann 2003). For NN, we used multilayer perceptron (MLP) classifier. The splitting ratio of training and testing set is 70% versus 30%. We used the mature ML Python library *Scikit-Learn* (Pedregosa et al. 2011) (version 0.19.1), with the best hyperparameters from fine-tuning for the 9 ML algorithms to compare the ML algorithms' performance. The performance comparison of the nine ML algorithms plus ensemble max-voting of the nine ML algorithms is presented in Table 1.

From Table 1, it is obvious that GBC (100%, 86%), RF (98.14%, 85.67%), and max-voting classifier (89.29%, 84.33%) ranked as top 3 trained models in terms of training and testing accuracy. Among the top 3, max-voting classifier took the least time to train the model (0.95s), about half of the time GBC required (2.033s), and RF took 1.38s. As introduced in Sect. 2, DT suffers from overfitting (90.86%, 79.33%)—actually, all the models in italic in Table 1 encountered different degrees of overfitting. GBC does not scale well

(a) Top 3 TensorFlow output: dam, dike, dyke; alp; valley, vale

(b) Top 3 TensorFlow output: maze, labyrinth; wing; car mirror

(c) Top 3 TensorFlow output: lakeside, lakeshore; boathouse; castle

Fig. 4 Manually labeled flooding CAP images examples. The sub-figure caption gives the top 3 TensorFlow outputs for the corresponding aerial image. Annotators do not see the TensorFlow outputs when they manually label the images as flooded or not

because it needs to be trained sequentially, not in parallel. We thus select max-voting classifier as our final model to classify the larger set of unlabeled aerial images as flooded or not, because it is obvious that max-voting is the best model in terms of training an testing score (thus without overfitting) and that it requires less training time compared with other trained models that has good training and testing accuracy (i.e., GBC and RF).

4.6 Ensemble max-voting classification model evaluation

From Table 1 and the model comparison and selection analysis in Sect. 4.5, we choose the ensemble max-voting classification model as the final model to automatically classify the rest of unlabeled 21,891 aerial images as flooded or not. To further evaluate whether the max-voting model can be generalized to new unseen data beyond the labeled data from the training and testing set, we randomly selected 500 aerials images from the unlabeled 21,891 images and man-

Table 1 Comparison of nine commonly used ML algorithms performance and the ensemble max-voting classifier combined all the nine ML algorithms

ML model	Training accuracy (%)	Testing accuracy (%)	Training time (s)	Trained model size
GBC	***100.00***	***86.00***	**2.033**	905 KB
RF	***98.14***	***85.67***	**1.380**	128 KB
DT	*90.86*	*79.33*	0.010	5.3 KB
NB	78.71	77.67	0.022	1.3 KB
LDA	78.14	76.67	0.008	1.7 KB
LR	77.14	77.33	1.078	945 bytes
KNN	*100.00*	*64. 67*	0.001	145 KB
SVM	77.29	77.67	96.744	74 KB
NN	59.71	64.67	1.260	42 KB
Max-voting classifier	**89.29**	**84.33**	**0.950**	2.5 MB

GBC gradient boosting classifier, *RF* random forest, *DT* decision tree, *NB* Naive Bayes, *LDA* linear discriminant analysis, *LR* logistic regression, *SVM* support vector machine, *KNN* K-nearest neighbors, *NN* neural network

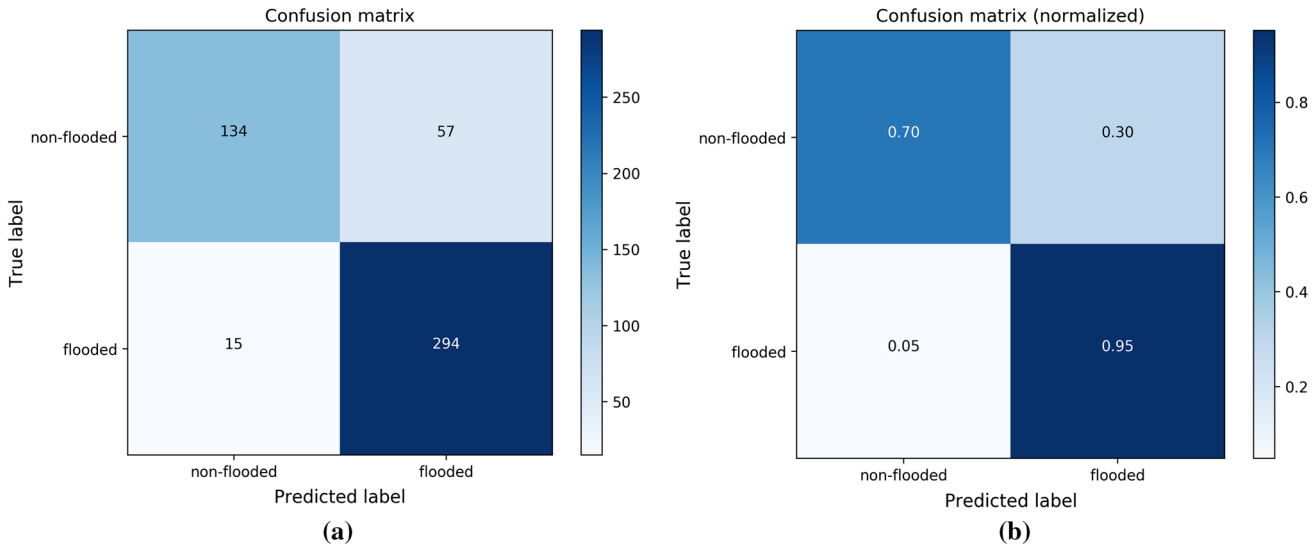


Fig. 5 Confusion matrices for max-voting classifier

ually checked whether the prediction using the max-voting model is correct or not. While performing the max-voting model evaluation using the 500 manually checked images, we used multiple metrics, in particular accuracy (85.6%), F1 (89.09%), precision (83.76%), and recall (95.15%), as well as non-normalized and normalized confusion matrices. The confusion matrices are shown in Fig. 5.

Three examples of predicted flooding aerial images, along with its corresponding top 3 TensorFlow outputs, are shown in Fig. 6. See Fig. 7 for the overview map of the aerial images classified as flooded or not, where Fig. 8 presents a zoomed-in detail.

5 Conclusion

Spatiotemporal big data can aid in monitoring the planet. To digest and transform this vast amount of data into actionable insights to help us live better, advanced methods and technologies are required. With the rapid development of computational power, machine learning and deep learning models can be trained using large sets of annotated data to automatically classify raw data (e.g., image and text) from different sources into pre-defined categories. GeoAI can allow us to observe environmental systems and how they are changing through time in a quick manner, by turning the data into useful information and using that information to generate actionable insights.

In this research, we propose a method that can leverage the integrated power of deep learning and machine learning to

(a) Top 3 TensorFlow output: dam, dike, dyke; boathouse; viaduct

(b) Top 3 TensorFlow output: valley, vale; cliff, drop, drop-off; rapeseed

(c) Top 3 TensorFlow output: lakeside, lakeshore; boathouse; castle

Fig. 6 Examples of predicted aerial images automatically classified as flooded, along with its corresponding top 3 TensorFlow outputs

construct an ensemble max-voting machine learning classifier with good performance from multiple machine learning algorithms using only a small set of manually labeled data. One important strength of our method, comparing with pure deep learning based methods, is that domain-specific meta-data relevant features that cannot be directly extracted from images themselves using deep learning, for example, geographical information (latitude and longitude) comes with aerial images, can be integrated into machine learning models. To demonstrate the effectiveness of the proposed methodology, we have used a flooding event as a test bed. The evaluation of the max-voting classifier on the case study using multiple metrics, including accuracy (85.6%), F1 score (89.09%), precision (83.76%), recall (95.15%), and confusion matrix (see Fig. 5), has demonstrated the effectiveness of our method.

This method can be applied not only to flooding events, because it is domain independent and can be applied to other domains. Examples include hurricanes, earthquakes, as well as feature extraction related to food security and transportation from aerial images, which is emphasized in the AI

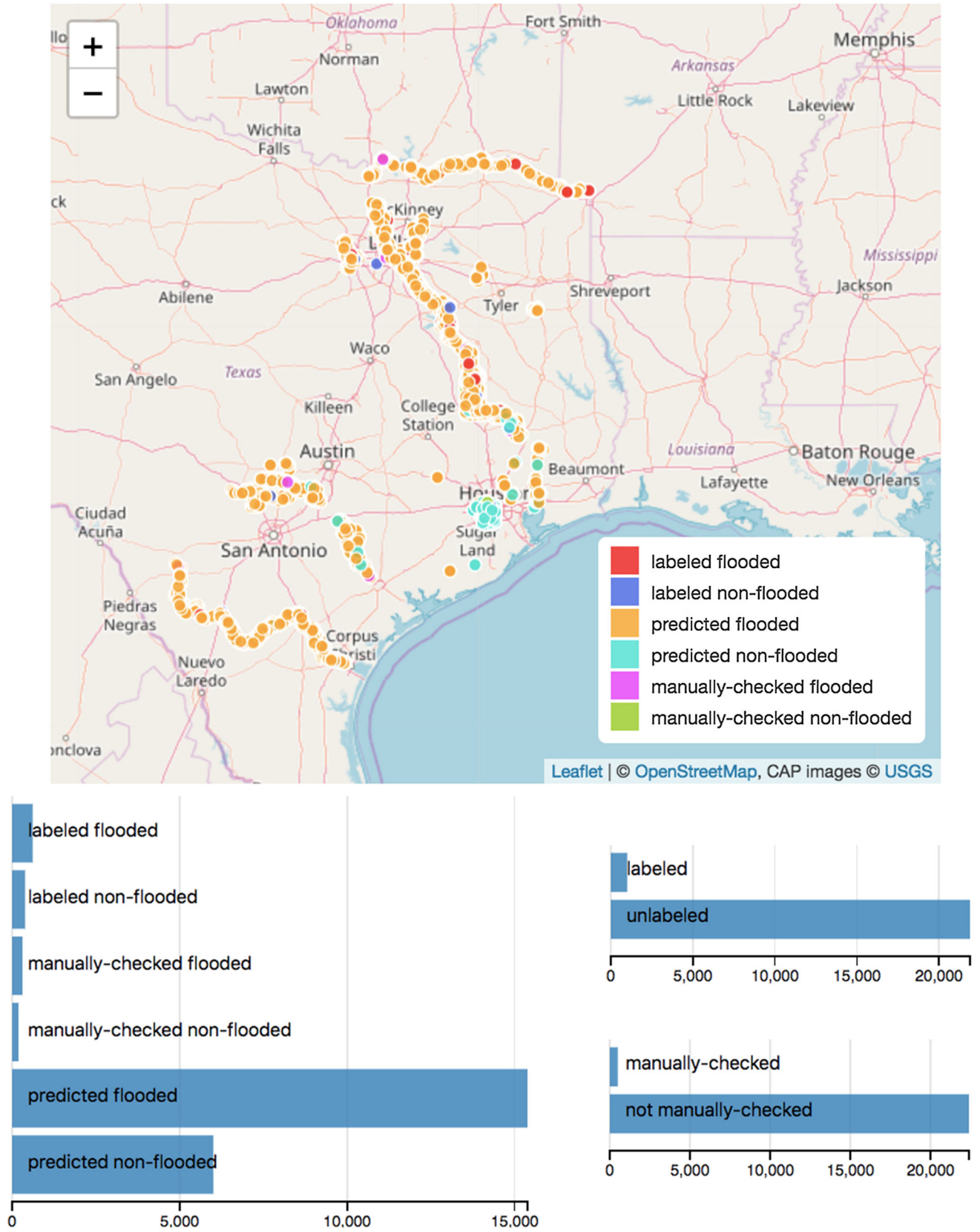


Fig. 7 Overview map of 22,891 aerial images classified as flooded or not, among which 1000 are manually labeled (613 flooded and 387 not flooded), and flooding status of 21,891 is automatically classified by our ensemble max-voting ML model that integrated multiple ML models (see Fig. 8 for a zoomed-in detail)

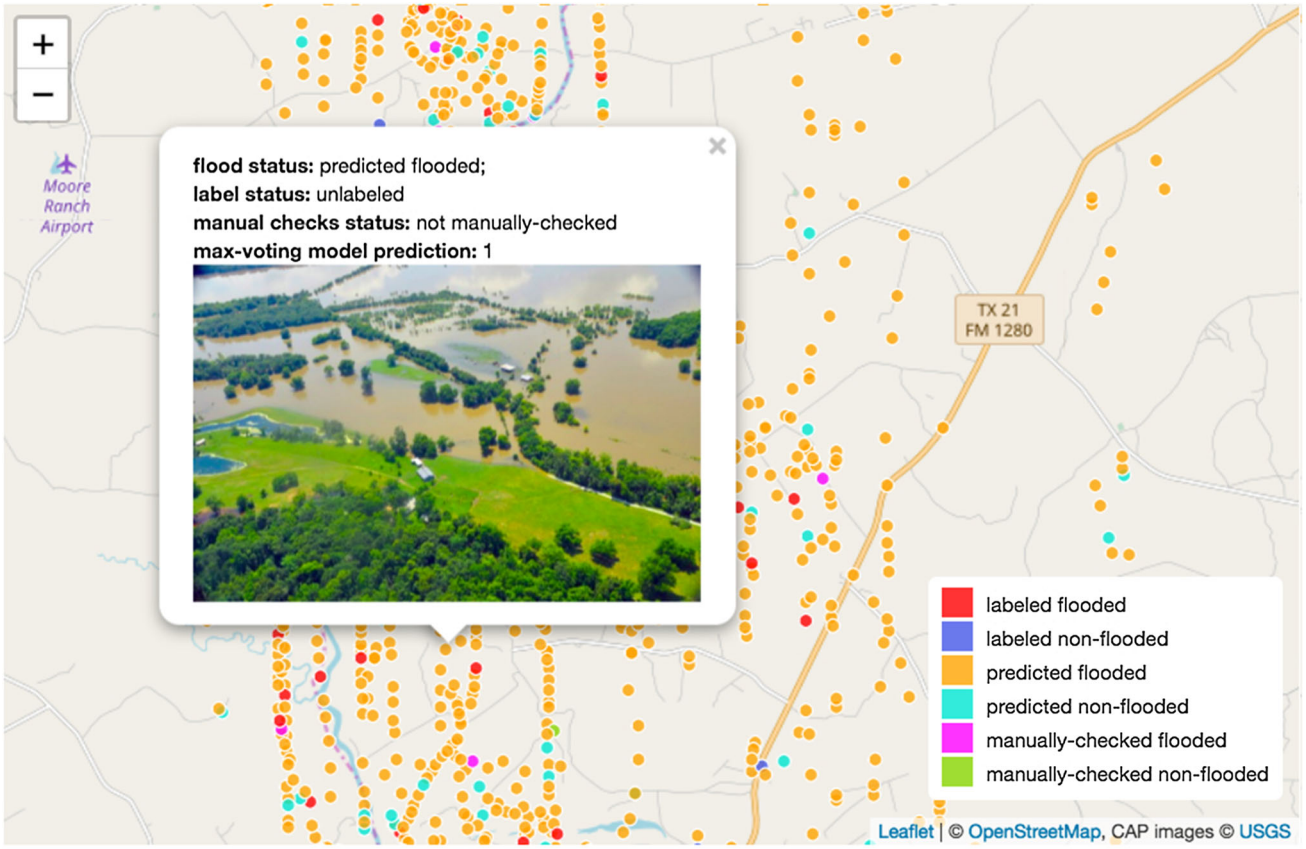


Fig. 8 A zoomed-in detail for Fig. 7. The value of max-voting model prediction for the demonstration aerial image (i.e., 1) means the trained max-voting model automatically classifies this aerial image as flooded

challenge by the World Bank, in collaboration with WeRobotics and OpenAerialMap.[2]

Acknowledgements This work was partially supported by the Office of Naval Research (ONR) award no. N00014-16-1-2543 (PSU no. 171570) and by the NVIDIA Corporation. We acknowledge Dr. Davide Del Vento from NCAR CISL and Dr. Chuck Pavloski at the Penn State Institute for CyberScience (ICS). The authors wish to thank Elena Sava for useful discussions and for providing the data and initial results relative to the Texas flood event.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

[2] Using AI For Good: A New Data Challenge To Use AI To Triage Natural Disaster Aerial Imagery https://www.forbes.com/sites/kalevleetaru/2018/01/20/using-ai-for-good-a-new-data-challenge-to-use-ai-to-triage-natural-disaster-aerial-imagery.

A Hyperparameter tuning settings and tuned results

In this appendix, we provide some details of the hyperparameter tuning process and results for the multiple machine learning algorithms introduced in Sect. 4.5. Appendix A.1 provides the hyperparameter settings including hyperparameter grids for each ML algorithms we tuned, and Appendix A.2 presents the optimized hyperparameters for each ML algorithm and how many percentage of accuracy each algorithm improved compared with its corresponding baseline model that uses the default hyperparameters in Scikit-Learn (version 0.19.1).

A.1 Hyperparameter settings for tuning multiple ML algorithms

While machine learning model parameters are learned during training (such as weights and bias in a neural network), hyperparameters need to be set before training by researchers or data scientists. Taking a neural network as an example, its hyperparameters include the number of hidden layers, number of neurons in each hidden layer, how many itera-

Table 2 Hyperparameter settings

ML model	Hyperparameter grid	Combinations	Fits
GBC	{ 'learning_rate': [0.001, 0.005, 0.01, 0.05, 0.1, 0.2], 'min_samples_split': [2, 4, 6, 8, 10, 20], 'min_samples_leaf': [2,4,6,8], 'max_depth': [5, 8, 10, 15, 20, 25], 'max_features': ['sqrt,' 'auto,' 'log2,' None], 'subsample': [0.5, 0.6, 0.8, 0.9, 1.0], 'n_estimators': [10, 50, 100, 150, 200] }	86,400	432,000
RF	{ 'n_estimators': [10, 20, 30, 40, 50, 60, 70, 75, 80, 85], 'max_depth': [5, 10, 15, 20, 25, 30, 40], 'min_samples_leaf': [2, 4, 6, 8, 10], 'max_features': ['sqrt,' 'auto,' 'log2,' None] }	1400	7000
DT	{ 'criterion': ['gini,' 'entropy'], 'max_depth': [None, 10, 15, 20, 25, 30, 35], 'min_samples_leaf': [2, 4, 6, 8, 10], 'max_features': ['auto,' 'sqrt,' None] }	210	1050
LR	{ 'penalty': ['l1,' 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 100] }	12	60
KNN	{ 'n_neighbors': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48], 'weights': ['uniform,' 'distance'], 'p': [1, 2] }	96	480
SVM	{ 'kernel': ['linear,' 'poly,' 'rbf,' 'sigmoid'], 'gamma': [10, 1, 0.1, 0.01, 0.001, 0.00001], 'C': [0.1, 1, 10, 100, 1000] }	120	600
NN	{ 'alpha': [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 1e-2]; 'hidden_layer_sizes': [(3,5), (5,10), (9,15), (25,), (50,), (100,), (200,)]; 'solver': ['lbfgs']; 'random_state': [None, 2, 4, 5, 7, 9]; 'max_iter': [100, 200, 300, 500]; 'activation': ['identity,' 'logistic,' 'tanh,' 'relu'] }	4032	20,160

tions, and what is activation function to use. Scikit-Learn has implemented a set of sensible default hyperparameters for all models, but these are not guaranteed to be optimal for a specific problem. The best hyperparameters are in general impossible to be pre-determined, and tuning hyperparameters of a model is where machine learning turns from science into trial-and-error-based engineering.

Hyperparameter tuning relies more on experimental results than theory, and thus, the best practice to determine the optimal settings is to run ML models on many different combinations of hyperparameters and evaluate the performance of each model. Evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning: *overfitting*. An overfit model may look impressive on the training set, but will most probably not be able to generalize to new unseen data in a real-world application. The standard procedure for hyperparameter optimization avoiding overfitting is through cross-validation.

In this research, we use 5-fold cross-validation while tuning the ML models (as introduced in Sect. A.2, 10-fold is a better setting. However, considering the expensive computation of tuning many combinations of hyperparameters, we chose 5-fold for the tuning process.) We also used all CPU cores (20 cores) available on our Linux Server. Table 2 provides the hyperparameter settings we used to search best parameter for each ML model using Scikit-Learn GridSearch. In Table 2, *Combinations* refers to the number of hyperparameter combinations and *Fits* refers to the number of model fits calculated based on the fold (in our case, the fold number = 5) used for cross-validation and the hyperparameter combinations.

Note that we used nine algorithms in total in Sect. 4.5, but we just tune seven ML models here. The reasons are as follows: for NB, there is no hyperparameter to be tuned; LDA has a closed-form solution and therefore has no hyperparameters to be tuned either.

A.2 Hyperparameter tuning results and comparison with baseline ML models

Using the hyperparameter settings we provided in Table 2, we tuned the ML models, and Table 3 provides the best parameters from our hyperparameter tuning and also the performance improvement compared with corresponding baseline ML models that used default hyperparameter settings.

In Table 3, in the training accuracy and testing accuracy columns, left value refers to accuracy score of the best model and right refers to that of baseline model. We can see that almost all ML models have better performance when using best hyperparameters from tuning, and some models are improved substantially (e.g., SVM and NN).

While comparing the model improvement using best hyperparameters from tuning against baseline line model, we used 10-fold cross-validation. The main reason we set k as 10-fold is based on conclusion in the literature, as Witten et al. (2011) introduced, tests on different datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error.

Table 3 Hyperparameter tuning results

ML model	Time	Best parameters	Training accuracy (best, baseline) (%)	Testing accuracy (best, baseline) (%)	Improvement of training accuracy (%)	Improvement of testing accuracy (%)
GBC	191.4 m	{ 'learning_rate': 0.1, 'min_samples_split': 20, 'max_depth': 25, 'min_samples_leaf': 8, 'subsample': 0.9, 'max_features': None, 'n_estimators': 150 }	87.71, 85.57	86.67, 85.00	2.50	1.96
RF	1.3 m	{ 'max_features': None, 'n_estimators': 20, 'max_depth': 15, 'min_samples_leaf': 2 }	86.43, 82.86	86.00, 81.33	4.31	5.74
DT	0.9 s	{ 'max_depth': 25, 'min_samples_leaf': 10, 'max_features': None, 'criterion': 'entropy' }	83.57, 80.86	82.00, 77.67	3.36	5.58
LR	6.6 s	{ 'penalty': 'l1,' 'C': 100 }	75.00, 66.29	74.33, 61.00	13.15	21.86
KNN	1.3 s	{ 'weights': 'distance,' 'n_neighbors': 18, 'p': 1 }	61.00, 58.57	63.67, 57.00	4.15	11.70
SVM	137.1 m	{ 'C': 0.1, 'gamma': 10, 'kernel': 'linear' }	76.57, 59.71	78.67, 64.67	28.23	21.65
NN	83.9 m	{ 'activation': 'identity,' 'hidden_layer_sizes': (200,), 'solver': 'lbfgs,' 'alpha': 0.01, 'max_iter': 500, 'random_state': None }	58.57, 51.86	59.00, 49.33	12.95	19.59

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. OSDI 16:265–283

Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat 46(3):175–185

Amancio DR, Comin CH, Casanova D, Travieso G, Bruno OM, Rodrigues FA, da Fontoura Costa L (2014) A systematic comparison of supervised classifiers. PLoS ONE 9(4):e94–137

Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2001) Support vector clustering. J Mach Learn Res 2(Dec):125–137

Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc., Newton

Bishop MC (2006) Pattern recognition and machine learning. Springer, New York

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Buhmann MD (2003) Radial basis functions: theory and implementations, vol 12. Cambridge University Press, Cambridge

Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2(2):121–167

Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on machine learning. ACM, pp 161–168

Cervone G, Sava E, Huang Q, Schnebele E, Harrison J, Waters N (2016) Using Twitter for tasking remote-sensing data collection and damage assessment: 2013 Boulder flood case study. Int J Remote Sens 37(1):100–124

Daelemans W, Van den Bosch A (2005) Memory-based language processing. Cambridge University Press, Cambridge

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: IEEE conference on computer vision and pattern recognition. CVPR 2009, IEEE, pp 248–255

Domingos P (2012) A few useful things to know about machine learning. Commun ACM 55(10):78–87

Domingos P (2015) The master algorithm: how the quest for the ultimate learning machine will remake our world. Basic Books, New York

Domingos P, Pazzani M (1997) On the optimality of the simple Bayesian classifier under zero-one loss. Mach Learn 29(2–3):103–130

Dubitzky W, Granzow M, Berrar DP (2007) Fundamentals of data mining in genomics and proteomics. Springer, Berlin

Elman JL (1990) Finding structure in time. Cognit Sci 14(2):179–211

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232

Gislason PO, Benediktsson JA, Sveinsson JR (2006) Random forests for land cover classification. Pattern Recognit Lett 27(4):294–300

Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J et al (2017) Recent advances in convolutional neural networks. Pattern Recognit 77:354

Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction, 2nd edn. Springer, New York

Kamiński B, Jakubczyk M, Szufel P (2018) A framework for sensitivity analysis of decision trees. Cent Eur J Oper Res 26(1):135–159

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Li Z, Wang C, Emrich CT, Guo D (2018) A novel approach to leveraging social media for rapid flood mapping: a case study of the 2015 South Carolina floods. Cartogr Geogr Inf Sci 45(2):97–110

Liong CY, Foo SF (2013) Comparison of linear discriminant analysis and logistic regression for data classification. In: AIP conference proceedings, AIP, vol 1522, pp 1159–1165

Murthy SK (1998) Automatic construction of decision trees from data: a multi-disciplinary survey. Data Mining Knowl Discov 2(4):345–389

Ng AY, Jordan MI (2002) On discriminative vs. generative classifiers: a comparison of logistic regression and Naive Bayes. In: Advances in neural information processing systems, pp 841–848

Opitz DW, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res (JAIR) 11:169–198

Panteras G, Cervone G (2018) Enhancing the temporal resolution of satellite-based flood extent generation using crowdsourced data for disaster monitoring. Int J Remote Sens 39(5):1459–1474

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12(Oct):2825–2830

Polikar R (2006) Ensemble based systems in decision making. IEEE Circuits Syst Mag 6(3):21–45

Press SJ, Wilson S (1978) Choosing between logistic regression and discriminant analysis. J Am Stat Assoc 73(364):699–705

Provost F, Kohavi R (1998) Glossary of terms. J Mach Learn 30(2–3):271–274

Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106

Rokach L (2010) Ensemble-based classifiers. Artif Intell Rev 33(1–2):1–39

Russell SJ, Norvig P, Canny JF, Malik JM, Edwards DD (2003) Artificial intelligence: a modern approach, vol 2. Prentice Hall, Upper Saddle River

Salzberg SL (1997) On comparing classifiers: pitfalls to avoid and a recommended approach. Data Mining Knowl Discov 1(3):317–328

Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. Inf Process Manag 45(4):427–437

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A, et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826

Wainer J (2016) Comparison of 14 different families of classification algorithms on 115 binary datasets. arXiv preprint arXiv:1606.00930

Weiss GM, Provost F (2003) Learning when training data are costly: the effect of class distribution on tree induction. J Artif Intell Res 19:315–354

Witten IH, Frank E, Hall MA (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann, Burlington

Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8(7):1341–1390

Xiao T, Xia T, Yang Y, Huang C, Wang X (2015) Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2691–2699

Yang L, MacEachren AM, Mitra P, Onorati T (2018) Visually-enabled active deep learning for (geo) text and image classification: a review. ISPRS Int J Geo-Inf 7(2):65

Zhu XX, Tuia D, Mou L, Xia GS, Zhang L, Xu F, Fraundorfer F (2017) Deep learning in remote sensing: a review. arXiv preprint arXiv:1710.03959