A Machine Learning Tutorial for Operational Meteorology. Part I: Traditional Machine Learning

RANDY J. CHASE,[a,b,c] DAVID R. HARRISON,[b,d,e] AMANDA BURKE,[b,c] GARY M. LACKMANN,[f] AND AMY MCGOVERN[a,b,c]

[a] *School of Computer Science, University of Oklahoma, Norman, Oklahoma*
[b] *School of Meteorology, University of Oklahoma, Norman, Oklahoma*
[c] *NSF AI Institute for Research on Trustworthy AI in Weather, Climate, and Coastal Oceanography, University of Oklahoma, Norman, Oklahoma*
[d] *Cooperative Institute for Severe and High-Impact Weather Research and Operations, University of Oklahoma, Norman, Oklahoma*
[e] *NOAA/NWS/Storm Prediction Center, Norman, Oklahoma*
[f] *Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University, Raleigh, North Carolina*

(Manuscript received 14 April 2022, in final form 26 May 2022)

ABSTRACT: Recently, the use of machine learning in meteorology has increased greatly. While many machine learning methods are not new, university classes on machine learning are largely unavailable to meteorology students and are not required to become a meteorologist. The lack of formal instruction has contributed to perception that machine learning methods are "black boxes" and thus end-users are hesitant to apply the machine learning methods in their everyday workflow. To reduce the opaqueness of machine learning methods and lower hesitancy toward machine learning in meteorology, this paper provides a survey of some of the most common machine learning methods. A familiar meteorological example is used to contextualize the machine learning methods while also discussing machine learning topics using plain language. The following machine learning methods are demonstrated: linear regression, logistic regression, decision trees, random forest, gradient boosted decision trees, naïve Bayes, and support vector machines. Beyond discussing the different methods, the paper also contains discussions on the general machine learning process as well as best practices to enable readers to apply machine learning to their own datasets. Furthermore, all code (in the form of Jupyter notebooks and Google Colaboratory notebooks) used to make the examples in the paper is provided in an effort to catalyze the use of machine learning in meteorology.

KEYWORDS: Radars/Radar observations; Satellite observations; Forecasting techniques; Nowcasting; Operational forecasting; Artificial intelligence; Classification; Data science; Decision trees; Machine learning; Model interpretation and visualization; Regression; Support vector machines; Other artificial intelligence/machine learning

1. Introduction

The mention and use of machine learning (ML) within meteorological journal articles is accelerating (Fig. 1; e.g., Burke et al. 2020; Hill et al. 2020; Lagerquist et al. 2020; Li et al. 2020; Loken et al. 2020; Mao and Sorteberg 2020; Muñoz-Esparza et al. 2020; Wang et al. 2020; Bonavita et al. 2021; Cui et al. 2021; Flora et al. 2021; Hill and Schumacher 2021; Schumacher et al. 2021; Yang et al. 2021; Zhang et al. 2021). With a growing number of published meteorological studies using ML methods, it is increasingly important for meteorologists to be well versed in ML. However, the availability of meteorology specific resources about ML terms and methods is scarce. Thus, this series of papers (total of two) aim to reduce the scarcity of meteorology specific ML resources.

While many ML methods are generally not new (i.e., published before 2002), there is a concern from ML developers that end users (i.e., non-ML specialists) may be hesitant or are concerned about trusting ML. However, early work in this space suggests that nontechnical explanations may be an important part of how end users perceive the trustworthiness of ML guidance (e.g., Cains et al. 2022). Thus, an additional goal of these papers is to enhance trustworthiness of ML methods through plain language discussions and meteorological examples.

In practice, ML models are often viewed as a *black box*, which could also be contributing to user hesitancy. These mystified feelings toward ML methods can lead to an inherent distrust with ML methods, despite their potential. Furthermore, the seemingly opaque nature of ML methods prevents ML forecasts from meeting one of the three requirements of a good forecast outlined by Murphy (1993): consistency. In short, Murphy (1993) explains that in order for a forecast to be good, the forecast must 1) be consistent with the user's prior knowledge, 2) have good quality (i.e., accuracy) and 3) be valuable (i.e., provide benefit). Plenty of technical papers demonstrate how ML forecasts can meet requirements 2 and 3, but as noted above if the ML methods are confusing and enigmatic, then it is difficult for ML forecasts to be consistent with a meteorologist's prior knowledge. This series of papers will serve as a reference for meteorologists in order to make the black box of ML more transparent and enhance user trust in ML.

This paper is organized as follows. Section 2 provides an introduction to all ML methods discussed in this paper and will

Denotes content that is immediately available upon publication as open access.

Corresponding author: Randy J. Chase, randychase@ou.edu

DOI: 10.1175/WAF-D-22-0070.1

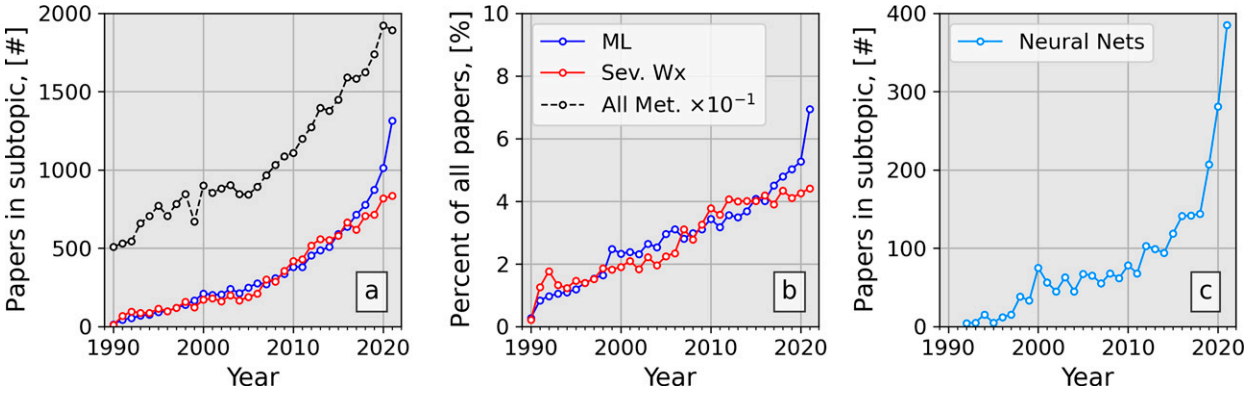

FIG. 1. Search results for the Meteorology and Atmospheric Science category when searching abstracts for machine learning methods and severe weather. Machine learning keywords searched were the following: linear regression, logistic regression, decision trees, random forest, gradient-boosted trees, support vector machines, *k*-means, *k*-nearest, empirical orthogonal functions, principal component analysis, self-organizing maps, neural networks, convolutional neural networks, and unets. Severe weather keywords searched were the following: tornadoes, hail, hurricanes, and tropical cyclones. (a) Counts of publications per year for all papers in the Meteorology and Atmospheric Science category (black line; reduced by one order of magnitude), machine learning topics (blue line), and severe weather topics (red line). (b) As in (a), but with the two subtopics normalized by the total number of Meteorology and Atmospheric Science papers. (c) Number of neural network papers (including convolutional and unets) published in Meteorology and Atmospheric sciences. All data are derived from Clarivate Web of Science.

define common ML terms. Section 3 discusses the general ML methods in context of a simple meteorological example, while also describing the end-to-end ML pipeline. Then, section 4 summarizes this paper and also discusses the topics of the next paper in the series.

2. Machine learning methods and common terms

This section will describe a handful of the most common ML methods. Before that, it is helpful to define some terminology used within ML. First, we define ML as any empirical[1] method where parameters are *fit* (i.e., learned) on a training dataset in order to *optimize* (e.g., minimize or maximize) a predefined *loss* (i.e., cost) function. Within this general framework, ML has two categories: *supervised* and *unsupervised* learning. Supervised learning are ML methods that are trained with prescribed input *features* and output *labels*. For example, predicting tomorrow's high temperature at a specific location where we have measurements (i.e., labels). Meanwhile, unsupervised methods do not have a predefined output label (e.g., self-organizing maps; Nowotarski and Jensen 2013). An example of an unsupervised ML task would be clustering all 500 mb geopotential height maps to look for unspecified patterns in the weather. This paper focuses on supervised learning.

The input features for supervised learning, also referred to as input data, predictors, or variables, can be written mathematically as the vector (matrix) $\mathbf{X}$. The desired output of the ML model is usually called the target, predictand or label, and is mathematically written as the scalar (vector) y. Drawing on the meteorological example of predicting tomorrow's high temperature, the input feature would be tomorrow's forecasted temperature from a numerical weather model (e.g., GFS) and the label would be tomorrow's observed temperature.

Supervised ML methods can be further broken into two subcategories: *regression* and *classification*. Regression tasks are ML methods that output a continuous range of values, like the forecast of tomorrow's high temperature (e.g., 75.0°F). Meanwhile classification tasks are characteristic of ML methods that classify data (e.g., will it rain or snow tomorrow). Reposing tomorrow's high temperature forecast as a classification task would be: "Will tomorrow be warmer than today?" This paper will cover both regression and classification methods. In fact, many ML methods can be used for both tasks.

All ML methods described here will have one thing in common: the ML method quantitatively uses the training data to optimize a set of weights (i.e., thresholds) that enable the prediction. These weights are determined either by minimizing the error of the ML prediction or maximizing a probability of a class label. The two different methods coincide with the regression and classification, respectively. Alternative names for error that readers might encounter in the literature are *loss* or *cost*.

Now that some of the common ML terms has been discussed, the following subsections will describe the ML methods. It will start with the simplest methods (e.g., linear regression) and move to more complex methods (e.g., support vector machines) as the sections proceed. Please note that the following subsections aim to provide an introduction and the intuition behind each method. An example of the methods being applied and helpful application discussion can be found in section 3.

a. Linear regression

An important concept in ML is when choosing to use ML for a task, one should start with the simpler ML models first.

[1] By "empirical" we mean any method that uses data as opposed to physics.

Occam's razor[2] tells us to prefer the simplest solution that can solve the task or represent the data. While this does not always mean the simplest ML model available, it does mean that simpler models should be tried before more complicated ones (Holte 1993). Thus, the first ML method discussed is linear regression, which has a long history in meteorology (e.g., Malone 1955) and forms the heart of the model output statistics product (i.e., MOS; Glahn and Lowry 1972) that many meteorologists are familiar with. Linear regression is popular because it is a simple method that is also computationally efficient. At its simplest form, linear regression approximates the value you would like to predict ($\hat{y}$) by fitting weight terms (w_i) in the following equation:

$$\hat{y} = \sum_{i=0}^{i=D} w_i x_i. \tag{1}$$

The first predictor (x_0) is always 1 so that w_0 is a bias term, allowing the function to move from the origin as needed. The term D is the number of features for the task.

As noted before, with ML, the objective is to find w_i such that a user-specified loss function (i.e., error function) is minimized. The most common loss function for traditional linear regression is the residual summed squared error (RSS):

$$\text{RSS} = \sum_{j=1}^{N} (y_j - \hat{y}_j)^2, \tag{2}$$

where y_j is a true data point, $\hat{y}_j$ is the predicted data point, and N is the total number of data points in the training dataset. A graphical example of a linear regression and its residuals is shown in Fig. 2. Linear regression using residual summed squared error can work very well and is a fast learning algorithm, so we suggest it as a baseline method before choosing more complicated methods. The exact minimization method is beyond the scope of this paper, but know that the minimization uses the slope (i.e., derivative) of the loss function to determine how to adjust the trainable weights. If this sounds familiar, that is because it is the same minimization technique learned in most first year college calculus classes and is a similar technique to what is used in data assimilation for numerical weather prediction (cf. Chapter 5 and section 10.5 in Kalnay 2002; Lackmann 2011). The concept of using the derivative to find the minimum is repeated throughout most ML methods given there is often a minimization (or maximization) objective.

Occasionally datasets can contain irrelevant or noisy predictors which can cause instabilities in the learning. One approach to address this is to use a modified version of linear regression known as ridge regression (Hoerl and Kennard 1970), which minimizes both the summed squared error (like before) and the sum of the squared weights called an L_2 penalty. Mathematically, the new loss function can be described as

$$\text{RSS}_{\text{ridge}} = \sum_{j=1}^{N} (y_j - \hat{y}_j)^2 + \lambda \sum_{i=0}^{D} w_i^2. \tag{3}$$

[2] https://en.wikipedia.org/wiki/Occam\%27s_razor.

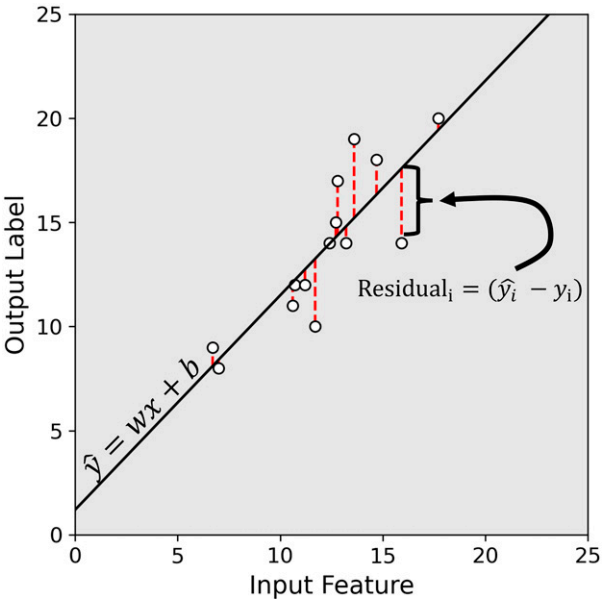

FIG. 2. A visual example of linear regression with a single input predictor. The x axis is a synthetic input feature, and the y axis is a synthetic output label. The solid black line is the regression fit, and the red dashed lines are the residuals.

Here, λ (which is ≥ 0) is a user-defined parameter that controls the weight of the penalty. Likewise, another modified version of linear regression is lasso regression (Tibshirani 1996) which minimizes the sum of the absolute value of the weights. This penalty to learning is also termed an L_1 penalty. The lasso loss function mathematically is

$$\text{RSS}_{\text{lasso}} = \sum_{j=1}^{N} (y_j - \hat{y}_j)^2 + \lambda \sum_{i=0}^{D} |w_i|. \tag{4}$$

Both lasso and ridge encourage the learned weights to be small but in different ways. The two penalties are often combined to create the elastic-net penalty (Zou and Hastie 2005):

$$\text{RSS}_{\text{elastic}} = \sum_{j=1}^{N} (y_j - \hat{y}_j)^2 + \lambda \sum_{i=0}^{D} [\alpha w_i^2 + (1 - \alpha)|w_i|]. \tag{5}$$

In general, the addition of components to the loss function, like described in Eqs. (3)–(5), is known as *regularization* and is found in other ML methods. Some recent examples of papers using linear regression include subseasonal prediction of tropical cyclone parameters (Lee et al. 2020), relating mesocyclone characteristics to tornado intensity (Sessa and Trapp 2020) and short term forecasting of tropical cyclone intensity (Hu et al. 2020).

b. *Logistic regression*

As a complement to linear regression, the first classification method discussed here is logistic regression. Logistic regression is an extension from linear regression in that it uses the same functional form of Eq. (1). The differences lie in how

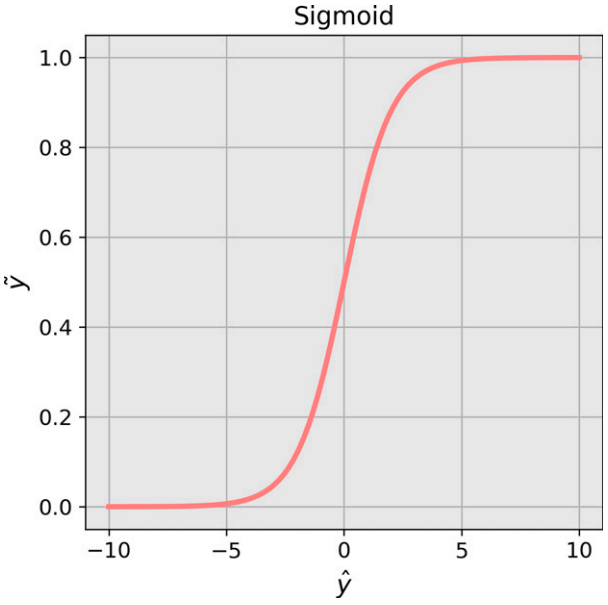

FIG. 3. A graphical depiction of the sigmoid function [Eq. (6)]. The x axis is the predicted label value, while the y axis is the now scaled value.

the weights for Eq. (1) are determined and a minor adjustment to the output of Eq. (1). More specifically, logistic regression applies the sigmoid function (Fig. 3) to the output of Eq. (1) defined as follows:

$$S(\hat{y}) = \frac{1}{1 + e^{-\hat{y}}}. \tag{6}$$

Large positive values into the sigmoid results in a value of 1 while large negative values result in a value of 0. Effectively, the sigmoid scales the output of Eq. (1) to a range from 0 to 1, which then can be interpreted like a probability. For the simplest case of classification involving just two classes (e.g., rain or snow), the output of the sigmoid can be interpreted as a probability of either class (e.g., rain or snow). The output probability then allows for the classification to be formulated as the w_i that maximizes the probability of a desired class. Mathematically, the classification loss function for logistic regression can be described as

$$\text{loss} = \sum_{i=0}^{i=D} -y_i \log\left[S(\hat{y})\right] + (1 - y_i)\log\left[1 - S(\hat{y})\right]. \tag{7}$$

Like before for linear regression, the expression in Eq. (7) is minimized using derivatives. If the reader is interested in more information on the mathematical techniques of minimization they can find more information in chapter 5 of Kalnay (2002).

Logistic regression has been used for a long time within meteorology. One of the earliest papers using logistic regression showed skill in predicting the probability of hail greater than 1.9 cm (Billet et al. 1997), while more recent papers have used logistic regression to identify storm mode (Jergensen et al. 2020), subseasonal prediction of surface temperature (Vigaud et al. 2019) and predict the transition of tropical cyclones to extratropical cyclones (Bieli et al. 2020).

c. *Naïve Bayes*

An additional method to do classification is known as naïve Bayes (Kuncheva 2006), which is named for its use of Bayes's theorem and can be written as the following:

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}. \tag{8}$$

In words, Eq. (8) is looking for the probability of some label y (e.g., snow), given a set of input features x [$P(y|x)$; e.g., temperature]. This probability can be calculated from knowing the probability of the label y occurring in the dataset [$P(y)$; e.g., how frequent it snows] times the probability of the input features given it belongs to the class y [$P(x|y)$; e.g., how frequently is it 32°F when it is snowing], divided by the probability of the input features [$P(x)$]. The *naïve* part of the naïve Bayes algorithm comes from assuming that all input features x, are independent of one another and the term $P(x|y)$ can be modeled by an assumed distribution (e.g., normal distribution) with parameters determined from the training data. While these assumptions are often not true, the naïve Bayes classifier can be skillful in practice. A few simplification steps results in the following:

$$\hat{y} = \text{argmax}\left\{\log\left[P(y)\right] + \sum_{i=0}^{N} \log\left[P(x_i|y)\right]\right\}. \tag{9}$$

Again in words, the predicted class ($\hat{y}$) from naïve Bayes is the classification label (y) such that the sum of the log of the probability of that classification [$P(y)$] and the sum of log of all the probabilities of the specific inputs given the classification [$P(x_i|y)$] is maximized. To help visualize the quantity $P(x_i|y)$, a graphical example is shown in Fig. 4. This example uses surface weather measurements from a station near Marquette, Michigan, where data were compiled when it was raining and snowing. Figure 4 shows distribution of air temperature (i.e., an input feature) given the two classes (i.e., rain versus snow). To get $P(x_i|y)$, we need to assume an underlying distribution function. The common assumed distribution with naïve Bayes is the normal distribution:

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x - \mu}{\sigma}\right)\right], \tag{10}$$

where μ is the mean and σ is the standard deviation of the training data. While the normal distribution assumption for the temperature distribution in Fig. 4 is questionable due to thermodynamic constraints that *lock* the temperature at 32°F (i.e., latent cool/heating), naïve Bayes can still have skill. Initially, it might not seem like any sort of weights/biases are being fit like the previously mentioned methods (e.g., logistic regression), but μ and σ are being *learned* from the training data. If performance from the normal distribution is poor,

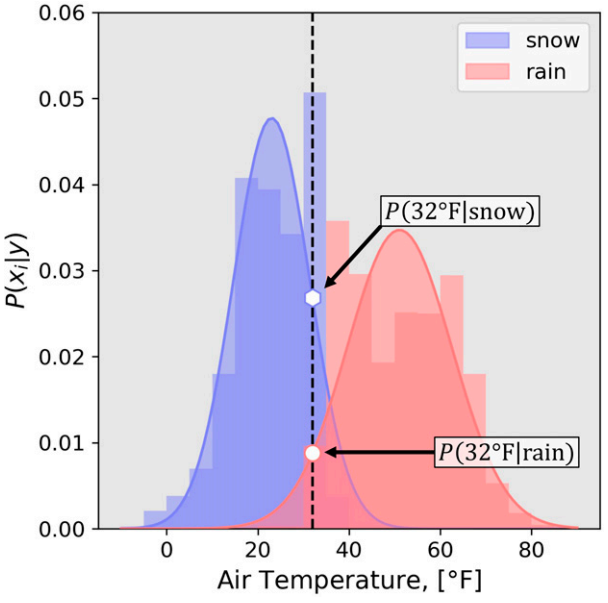

FIG. 4. Visualizing the probability of an input feature given the class label. This example is created from 5-min weather station observations from near Marquette, MI (years included: 2005–20). The precipitation phase was determined by the present weather sensor. The histogram is the normalized number of observations in that temperature bin, while the smooth curves are the normal distribution fit to the data. Red is for raining instances and blue is for snowing instances.

other distributions can be assumed, like a multinomial or a Bernoulli distribution.

A popular use of naïve Bayes classification in the meteorological literature has been the implementation of ProbSevere (e.g., Cintineo et al. 2014, 2018, 2020) which uses various severe storm parameters and observations to classify the likelihood of any storm becoming severe in the next 60 min. Additional examples of naïve Bayes classifiers in meteorology have been used for identifying tropical cyclone secondary eyewall formation from microwave imagery (Kossin and Sitkowski 2009), identifying anomalous propagation in radar data (Peter et al. 2013) and precipitation type (e.g., convective/stratiform) retrievals from geostationary satellites (Grams et al. 2016).

d. *Trees and forests*

Decision trees are based on a decision making method that humans have been using for years: flow charts, where the quantitative decision points within the flowchart are learned automatically from the data. Early use of decision trees in meteorology (e.g., Chisholm et al. 1968) actually predated the formal description of the decision tree algorithm (Breiman 1984; Quinlan 1993; Breiman 2001). Since then, tree-based methods have grown in popularity and have been demonstrated to predict a variety of complex meteorological phenomena. Topics include the following: aviation applications (e.g., Williams et al. 2008a,b; Williams 2014; Muñoz-Esparza et al. 2020), severe weather (e.g., Gagne et al. 2009, 2013; McGovern et al. 2014; Mecikalski et al. 2015; Lagerquist et al. 2017; Gagne et al. 2017; Czernecki et al. 2019; Burke et al. 2020; Hill et al. 2020; Loken et al. 2020; Gensini et al. 2021; Flora et al. 2021; Loken et al. 2022), solar power (e.g., McGovern et al. 2015), precipitation (e.g., Elmore and Grams 2016; Herman and Schumacher 2018b,a; Taillardat et al. 2019; Loken et al. 2020; Wang et al. 2020; Mao and Sorteberg 2020; Li et al. 2020; Hill and Schumacher 2021; Schumacher et al. 2021), satellite and radar retrievals (e.g., Kühnlein et al. 2014; Conrick et al. 2020; Yang et al. 2021; Zhang et al. 2021), and climate-related topics (e.g., Cui et al. 2021).

To start, we will describe decision trees in context of a classification problem. The decision tree creates splits in the data (i.e., decisions) that are chosen such that either the Gini impurity value or the entropy value decreases after the split. Gini impurity is defined as

$$\text{Gini} = \sum_{i=0}^{i=k} p_i(1 - p_i), \tag{11}$$

where p_i is the probability of class i (i.e., the number of data points labeled class i divided by the total number of data points). While entropy is defined as

$$\text{entropy} = \sum_{i=0}^{i=k} p_i \log_2(p_i). \tag{12}$$

Both functions effectively measure how similar the data point labels are in each one of the groupings of the tree after some split in the data. Envision the flowchart as a tree. The decision is where the tree branches into two directions, resulting in two separate leaves. The goal of a decision tree is to choose the branch that results in a leaf having a minimum of Gini or entropy. In other words, the data split would ideally result in two subgroups of data where all the labels are the same within each subgroup. Figure 5 shows both the Gini impurity and entropy for a two class problem. Consider the example of classifying winter precipitation as rain or snow. From some example surface temperature dataset, the likely decision threshold would be near 32°F, which would result in the subsequent two groupings of data point labels (i.e., snow/rain) having a dominant class label (i.e., fraction of class k is near 0 or 1) and thus having a minimum of entropy or Gini (i.e., near 0). The actual output of this tree could be either the majority class label, or the ratio of the major class (i.e., a probabilistic output).

While it is helpful to consider a decision tree with a single decision, also known as a tree with a depth of 1, the prediction power of a single decision is limited. A step toward more complexity is to include increasing depth (i.e., more decisions/branches). To continue with the rain/snow example from the previous paragraph, we could include a second decision based on measured wet bulb temperature. A tree with depth two will likely have better performance, but the prediction power is still somewhat limited.

An additional step to increase the complexity of decision trees, beyond including more predictors, is a commonly used method in meteorology: ensembles. While it might not be

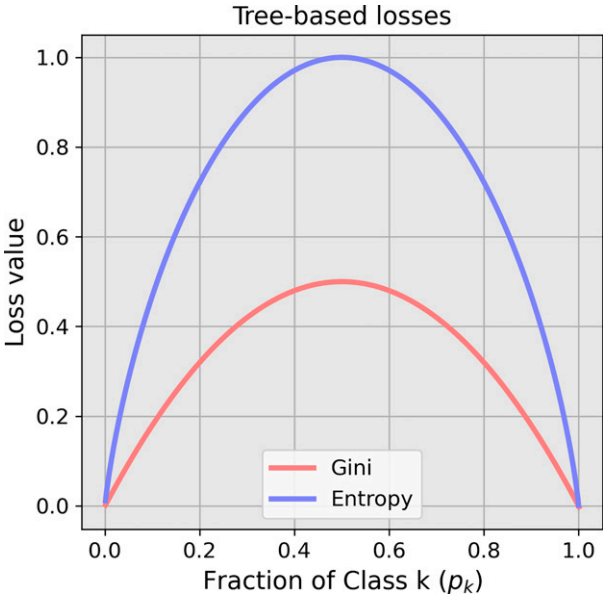

FIG. 5. A visual representation of the two functions that can be used in decision trees for classification, entropy (blue) and Gini impurity (red).

clear here, decision trees become over-fit (i.e., work really well for training data, but perform poorly on new data) as the depth of the tree increases. An alternative approach is to use an ensemble of trees (i.e., a forest). Using an ensemble of trees forms the basis of two additional tree based methods: random forests (Breiman 2001) and gradient boosted decision trees (Friedman 2001).

Random forests are a collection of decision trees that are trained on random subsets of data and random subsets of input variables from the initial training dataset. In other words, the mathematics are exactly the same for each tree, the decisions still aim to minimize the loss (e.g., entropy), but each tree is given a different random subset of data sampled from the original dataset with replacement. Gradient boosted decision trees are an ensemble of trees that instead of training multiple trees on random subsets (i.e., random forest), each tree in the ensemble is successively trained on the remaining error from the previous trees. To put it another way, rather than minimizing the total error on random trees, the reduced error from the first decision tree is now minimized on the second tree, and the reduced error from trees one and two is then minimized on the third tree and so on. To come up with a single prediction out of the ensemble of trees, the predictions can be combined through a voting procedure (i.e., count up the predicted classes of each tree) or by taking the average probabilistic output from each tree. Random forests can use either method, while gradient boosted trees are limited to the voting procedure.

While the discussion here has been centered on classification for the tree-based methods, they can be used for regression as well. The main alteration to the decision tree method to convert to a regression-based problem is the substitution of the loss function [i.e., Eq. (11) and (12)]. For example, a common loss function for random forest for regression and gradient boosted regression is the same loss function as linear regression described in the previous section [e.g., Eq. (2)], the residual summed squared error.

e. *Support vector machines*

A support vector machine (commonly referred to as SVM; Vapnik 1963) is an ML method similar to linear and logistic regression. The idea is that a support vector machine uses a linear boundary to do its predictions, which has a similar mathematical form but written differently to account to vector notation. The equation is

$$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b, \tag{13}$$

where $\mathbf{w}$ is a vector of weights, $\mathbf{x}$ is a vector of input features, b is a bias term, and $\hat{y}$ is the regression prediction. In the case of classification, only sign of the right side of Eq. (13) is used. This linear boundary can be generalized beyond two-dimensional problems (i.e., two input features) to three-dimensions where the decision boundary is called a plane, or any higher-order space where the boundary is called a hyperplane. The main difference between linear methods discussed in sections 2a and 2b and support vector machines is that support vector machines include margins to the linear boundary. Formally, the margin is the area between the linear boundary and the closest training data point for each class label (e.g., closest rain data point and closest snow data point). This is shown schematically with a synthetic dataset in Fig. 6a. While this is an ideal case, usually classes overlap (Fig. 6b), but support vector machines can still handle splitting the classes. The optimization task for support vector machines is stated as the following: Find $\mathbf{w}^{\mathrm{T}}$ such that the margin is maximized. In other words, support vector machines aim to maximize the distance between the two closest observations on either side of the hyperplane. Mathematically, the margin distance is described as

$$\text{margin} = \frac{1}{\mathbf{w}^{\mathrm{T}}\mathbf{w}}. \tag{14}$$

Like before, the maximization is handled by numerical techniques to optimize the problem, but the resulting solution will be the hyperplane with the largest separation between the classes. A powerful attribute of the support vector machine method is that it can be extended to additional mathematical formulations for the boundary, for example a quadratic function. Thus, the person using support vector machines can decide which function would work best for their data. Recent applications of support vector machines in meteorology include the classification of storm mode (Jergensen et al. 2020), hindcasts of tropical cyclones (Neetu et al. 2020), and evaluating errors with quantitative precipitation retrievals in the United States (Kurdzo et al. 2020).

3. Machine learning application and discussion

This section will discuss the use of all ML methods with a familiar use-case: thunderstorms. Specifically, this section will show two ML applications derived from popular meteorological datasets: radar and satellite. The particular data used are

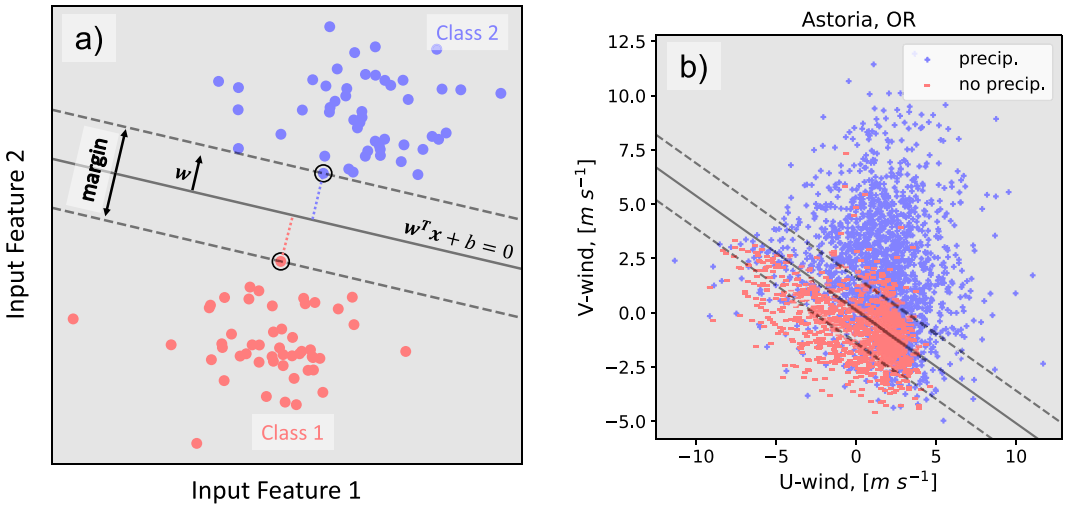

FIG. 6. Support vector machine classification examples. (a) Ideal (synthetic) data where the x and y axis are both input features, while the color designates what class each point belongs to. The decision boundary learned by the support vector machine is the solid black line, while the margin is shown by the dashed lines. (b) A real world example using NAM 1800 UTC forecasts of U and V wind and tipping-bucket measurements of precipitation. Blue plus markers are raining instances, and the red minus signs are non-raining instances. Black lines are the decision boundary and margins.

from the Storm Event Imagery dataset (SEVIR; Veillette et al. 2020), which contains over 10 000 storm events from between 2017 and 2019. Each event spans four hours and includes measurements from both *GOES-16* and NEXRAD. An example storm event and the five measured variables—red channel visible reflectance (0.64 μm; channel 2), midtropospheric water vapor brightness temperature (6.9 μm; channel 9), clean infrared window brightness temperature (10.7 μm; channel 13), vertically integrated liquid (VIL; from NEXRAD), and Geostationary Lightning Mapper (GLM) measured lightning flashes—are found in Fig. 7. In addition to discussing ML in context of the SEVIR dataset, this section will follow the general steps to using ML and contain helpful discussions of the best practices as well as the most common pitfalls.

a. *Problem statements*

The SEVIR data will be applied to two tasks: 1) Does this image contain a thunderstorm? and 2) How many lightning flashes are in this image? To be explicit, we assume the GLM observations are unavailable, and we need to use the other measurements (e.g., infrared brightness temperature) as features to estimate if there are lightning flashes (i.e., classification), and how many of them are there (i.e., regression). While both of these tasks might be considered redundant since we have GLM, the goal of this paper is to provide discussion on how to use ML as well as discussion on the ML methods themselves. That being said, a potential useful application of the trained models herein would be to use them on satellite sensors that do not have lightning measurements. For example, all generations of GOES prior to *GOES-16* did not have a lightning sensor collocated with the main sensor. Thus, we could potentially use the ML models trained here to estimate GLM measurements prior to *GOES-16* (i.e., November 2016).

b. *Data*

The first step of any ML project is to obtain data. Here, the data are from a public archive hosted on the Amazon web service. For information of how to obtain the SEVIR data as well as the code associated with this manuscript see the data availability statement. One major question at this juncture is as follows: "How much data are needed to do machine learning?" While there does not exist a generic number that can apply to all datasets, the idea is to obtain enough data such that one's training data are diverse. A diverse dataset is desired because any bias found within the training data would be encoded in the ML method (McGovern et al. 2021). For example, if a ML model was trained on only images where thunderstorms were present, then the ML model would likely not know what a non-lightning producing storm would look like and be biased. Diversity in the SEVIR dataset is created by including random images (i.e., no storms) from all around the United States (cf. Fig. 2 in Veillette et al. 2020).

After obtaining the data, it is vital to remove as much spurious data as possible before training because the ML model will not know how to differentiate between spurious data and high quality data. A common anecdote when using ML models is *garbage in, garbage out.* The SEVIR dataset has already gone through rigorous quality control, but this is often not the case with raw meteorological datasets. Two examples of quality issues that would likely be found in satellite and radar datasets are satellite artifacts (e.g., *GOES-17* heat pipe; McCorkel et al. 2019) and radar ground clutter (e.g., Hubbert et al. 2009). Cleaning and manipulating the dataset to get it ready for ML often takes a researcher 50%–80% of their

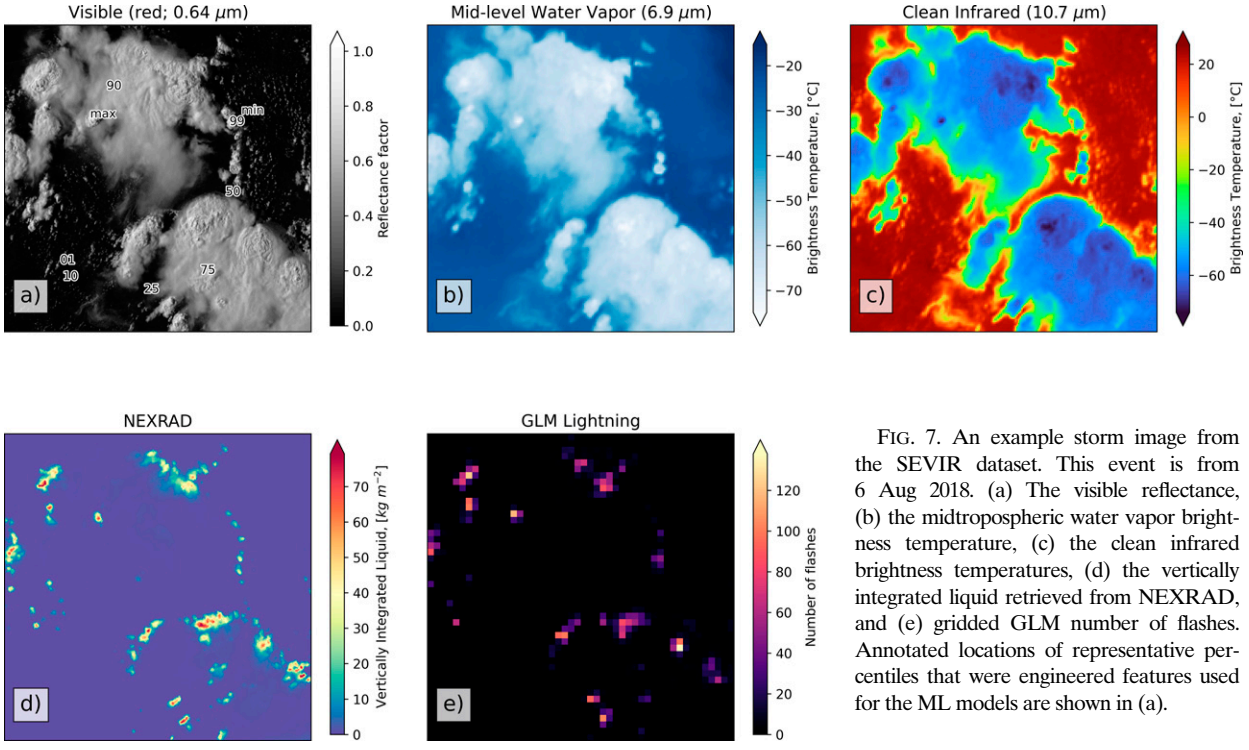

FIG. 7. An example storm image from the SEVIR dataset. This event is from 6 Aug 2018. (a) The visible reflectance, (b) the midtropospheric water vapor brightness temperature, (c) the clean infrared brightness temperatures, (d) the vertically integrated liquid retrieved from NEXRAD, and (e) gridded GLM number of flashes. Annotated locations of representative percentiles that were engineered features used for the ML models are shown in (a).

time.[3] Thus, do not be discouraged if cleaning one's datasets is taking a large amount of time because a high-quality dataset will be best for having a successful ML model.

Subsequent to cleaning the data, the next step is to engineer the inputs (i.e., features) and outputs (i.e., labels). One avenue to create features is to use every single pixel in the image as a predictor. While this could work, given the number of pixels in the SEVIR images (589 824 total pixels for one visible image) it is computationally impractical to train a ML model with all pixels. Thus, we are looking for a set of statistics than can be extracted from each image. For the generation of features, domain knowledge is critical because choosing meteorologically relevant quantities will ultimately determine the ML models skill. For the ML tasks presented in section 3a, information about the storm characteristics (e.g., strength) in the image would be beneficial features. For example, a more intense storm is often associated with more lightning. Proxies for estimating storm strength would be the magnitude of reflectance in the visible channel; how cold brightness temperatures in the water vapor and clean infrared channel are; and how much vertically integrated water there is. Thus, to characterize these statistics, we extract the following percentiles from each image and variable: 0, 1, 10, 25, 50, 75, 90, 99, and 100.

To create the labels the number of lightning flashes in the image are summed. For Problem Statement 1, an image is classified as containing a thunderstorm if the image has at least one flash in the last five minutes. For Problem Statement 2, the sum of all lightning flashes in the past five minutes within the image are used for the regression target.

Now that the data have been quality controlled and our features and labels have been extracted, the next step is to split that dataset into three independent subcategories named the *training*, *validation*, and *testing* sets. The reason for these three subcategories is because of the relative ease at which ML methods can "memorize" the training data. This occurs because ML models can contain numerous (e.g., hundreds, thousands, or even millions) learnable parameters, thus the ML model can learn to perform well on the training data but not generalize to other non-training data, which is called *over-fitting*. To assess how over-fit a ML model is, it is important to evaluate a trained ML model on data outside of its training data (i.e., validation and testing sets).

The training dataset is the largest subset of the total amount of data. The reason the training set is the largest is because the aforementioned desired outcome of most ML models is to generalize on wide variety of examples. Typically, the amount of training data is between 70% and 85% of the total amount of data available. The validation dataset, regularly 5%–15% of the total dataset, is a subset of data used to assess if a ML model is over-fit and is also used for evaluating best model configurations (e.g., the depth of a decision tree). These model configurations are also known as *hyper-parameters*. Machine learning models have numerous configurations and permutations that can be varied and could impact the skill of any

[3] https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html.

one trained ML model. Thus, common practice is to systematically vary the available hyper-parameter choices, also called a grid search, and then evaluate the different trained models based on the validation dataset. Hyper-parameters will be discussed in more detail later. The test dataset is the last grouping that is set aside to the very end of the ML process. The test dataset is often of similar size to the validation dataset, but the key difference is that the test dataset is used *after* all hyper-parameter variations have been concluded. The reason for this last dataset is because when doing the systematic varying of the hyper-parameters the ML practitioner is inadvertently tuning a ML model to the validation dataset. One will often choose specific hyper-parameters in such a way to achieve the best performance on the validation dataset. Thus, to provide a truly unbiased assessment of the trained ML model skill for unseen data, the test dataset is set aside and not used until after training all ML models.

It is common practice outside of meteorology (i.e., data science) to randomly split the total dataset into the three subsets. However, it is important to strive for independence of the various subsets. A data point in the training set should not be highly correlated to a data point in the test set. In meteorology this level of independence is often challenging given the frequent spatial and temporal autocorrelations in meteorologic data. Consider the SEVIR dataset. Each storm event has 4 h of data broken into 5-min time steps. For one storm event, there is a large correlation between adjacent 5-min samples. Thus, randomly splitting the data would likely provide a biased assessment of the true skill of the ML model. To reduce the number of correlated data points across subsets, time is often used to split the dataset. For our example, we choose to split the SEVIR data up by training on 1 January 2017–1 June 2019 and split every other week in the rest of 2019 into the validating and testing sets. This equates to a 72%, 13%, and 15% split for the training, validation, and test sets, respectively. In the event that the total dataset is small and splitting the data into smaller subsets creates less robust statistics, a resampling method known as *k-fold cross validation* (e.g., Bischl et al. 2012; Goodfellow et al. 2016) can be used. The SEVIR dataset was sufficiently large that we chose not to do *k*-fold cross validation, but a meteorological example using it can be found in Shield and Houston (2022).

c. *Training and evaluation*

1) CLASSIFICATION

As stated in section 3a, task 1 is to classify if an image contains a thunderstorm. Thus, the classification methods available to do this task are logistic regression, naïve Bayes, decision trees, random forest, gradient boosted trees, and support vector machines. To find an optimal ML model, it is often best to try all methods available. While this might seem like a considerable amount of additional effort, the ML package used in this tutorial (i.e., scikit learn[4]) uses the same syntax for all methods [e.g., method.fit(X, y), method.predict(X_{val})]. Thus, fitting all available methods does not require substantially more effort from the ML practitioner and will likely result in finding a best performing model.

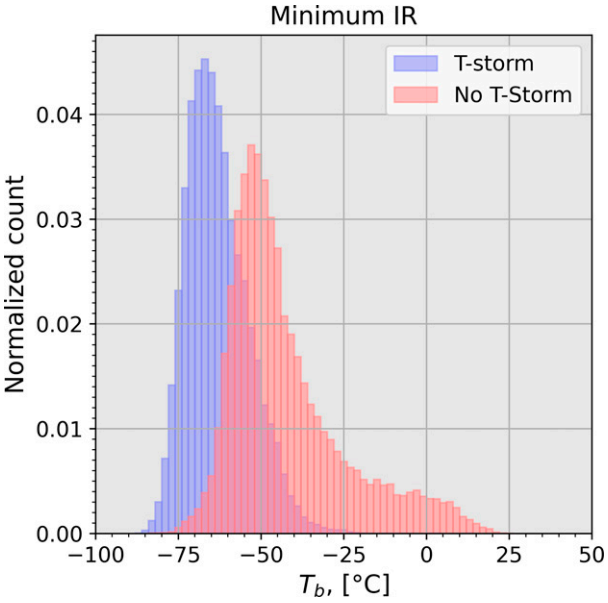

FIG. 8. The normalized distributions of minimum brightness temperature (T_b) from the clean infrared channel for thunderstorm images (blue; T-storm) and non-thunderstorm images (red; No T-storm).

To start off, all methods are initially trained using their default hyper-parameters in scikit-learn and just one input feature, the minimum infrared brightness temperature (T_b). We choose to use T_b because meteorologically it is a proxy for the depth of the storms in the domain, which is correlated to lightning formation (Yoshida et al. 2009). To assess the predictive power of this variable, the distributions of T_b for thunderstorms and no thunderstorms are shown in Fig. 8. As expected, T_b for thunderstorms shows more frequent lower temperatures than non-thunderstorm images. Training all methods using T_b achieves an accuracy of 80% on the validation dataset. While accuracy is a common and easy to understand metric, it is best to always use more than one metric when evaluating ML methods.

Another common performance metric for classification tasks is the area under the curve (AUC). More specifically the common area metric is associated with the receiver operating characteristics curve (ROC). The ROC curve is calculated from the relationship between the probability of false detection (POFD) and the probability of detection (POD). Both POFD and POD parameters are calculated from determining parameters within a contingency table which are the true positives (both the ML prediction and label say thunderstorm), false positives (ML prediction predicts thunderstorm, label has no thunderstorm), false negatives (ML prediction is no thunderstorm, label shows there is a thunderstorm) and true negatives (ML says no thunderstorm, label says no thunderstorm). The POFD and POD are defined by

[4] https://scikit-learn.org/stable/.

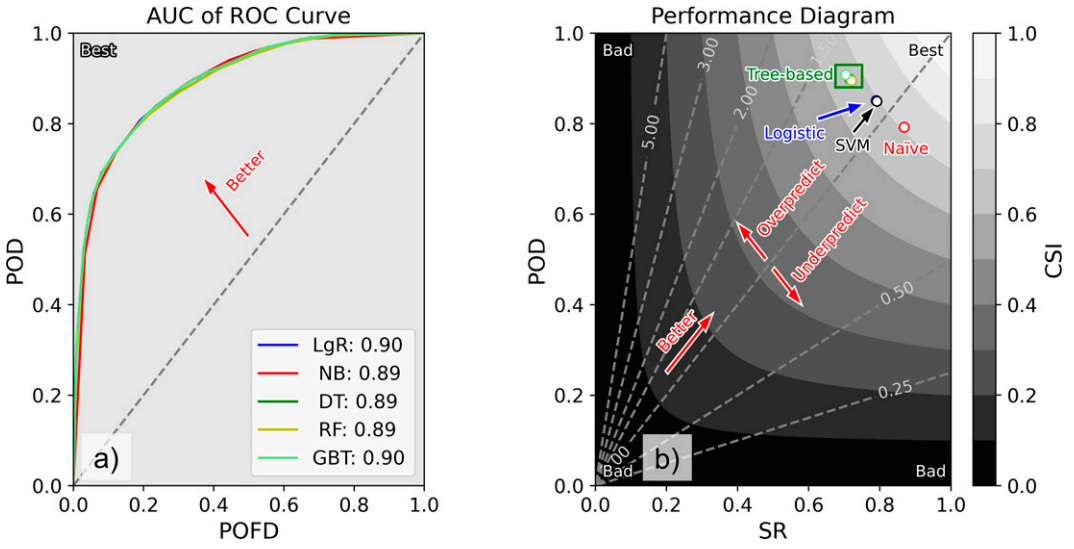

FIG. 9. Performance metrics from the simple classification (only using T_b). (a) Receiver operating characteristic (ROC) curves for each ML model (except support vector machines), logistic regression (LgR; blue), naïve Bayes (NB; red), decision tree (DT; geen), random forest (RF; yellow), and gradient boosted trees (GBT; light green). The area under the ROC curve is reported in the legend. (b) Performance diagram for all ML models [same colors as (a)]. Color fill is the corresponding CSI value for each success ratio–probability of detection (SR–POD) pair. Dashed contours are the frequency bias.

$$\text{POFD} = \frac{\text{FalsePositive}}{\text{TruePositive} + \text{FalsePositive}}, \tag{15}$$

$$\text{POD} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}. \tag{16}$$

All of the ML models, except support vector machines (as coded in sklearn), can provide a probabilistic estimation of the classification (e.g., this image is 95% likely to have lightning in it). When calculating the accuracy before, we assumed a threshold of 50% to designate what the ML prediction was. To get the ROC curve, the threshold probability is instead varied from 0% to 100%. The resulting ROC curves for all of the ML methods except support vector machines are shown in Fig. 9a. We see that for this simple one feature model, all methods are still very similar and have AUCs near 0.9 (Fig. 9a), which is generally considered good performance.[5]

An additional method for evaluating the performance of classification method is called a performance diagram (Fig. 9b; Roebber 2009). The performance diagram is also calculated from the contingency table, using the POD again for the y axis, but this time the x axis is the success ratio (SR) which is defined as

$$\text{SR} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}. \tag{17}$$

From this diagram, several things can be gleaned about the models' performance. In general, the top right corner is where "best" performing models are found. This area is characterized by models that capture nearly all events (i.e., thunderstorms), while not predicting a lot of false alarms (i.e., false positives). This corner is also associated with high values of critical success index (CSI; filled contours Fig. 9b), defined as

$$\text{CSI} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive} + \text{FalseNegative}}, \tag{18}$$

which is a metric that shows a model's performance without considering the true negatives. Not considering the true negatives is important because true negatives can dominate ML tasks in meteorology given the often rare nature of events with large impacts (e.g., floods, hail, tornadoes). The last set of lines on this diagram are the frequency bias contours (dashed gray lines Fig. 9b). These contours indicate if a model is overforecasting or underforecasting.

For the simple ML models trained, even though most of them have a similar accuracy and AUC, the performance diagram suggests their performance is indeed different. Consider the tree based methods (green box; Fig. 9b). They are all effectively at the same location with a POD of about 0.9 and a SR of about 0.75, which is a region that has a frequency bias of almost 1.5. Meanwhile the logistic regression, support vector machines and naïve Bayes methods are much closer to the frequency bias line of 1, while having a similar CSI as the tree based methods. Thus, after considering overall accuracy, AUC and the performance diagram, the best performing model would be the logistic regression, support vector machines, or naïve Bayes. At this junction, the practitioner has the option to consider if they want a slightly overforecasting

[5] No formal peer reviewed journal states this; it is more of a rule of thumb in machine learning practice.

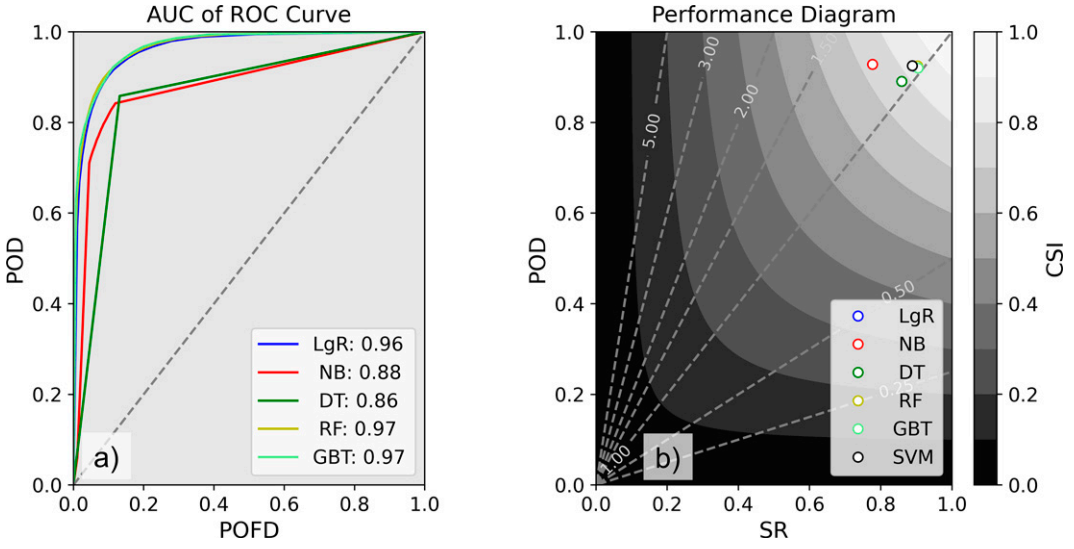

FIG. 10. As in Fig. 9, but now trained with all available predictors. The annotations from Fig. 9 have been removed.

system or a slightly underforecasting system. For the thunderstorm, no-thunderstorm task, there are not many implications for overforecasting or underforecasting. However, developers of a tornado prediction model may prefer a system that produces more false positives (overforecasting; storm warned, no tornado) than false negatives (underforecasting; storm not warned, tornado) as missed events could have significant impact to life and property. It should be clear that without going beyond a single metric, this differentiation between the ML methods would not be possible.

While the previous example was simple by design, we as humans could have used a simple threshold at the intersection of the two histograms in Fig. 8 to achieve similar accuracy (e.g., 81%; not shown). The next logical step with the classification task would be to use all available features. One important thing to mention at this step is that it is good practice to normalize input features. Some of the ML methods (e.g., random forest) can handle inputs of different magnitudes (e.g., CAPE is on the order of hundreds to thousands, but lifted index is on the order of one to tens), but others (e.g., logistic regression) will be unintentionally biased toward larger magnitude features if you do not scale your input features. Common scaling methods include min–max scaling and scaling your input features to have mean 0 and standard deviation of 1 (i.e., standard anomaly) which are defined mathematically as follows:

$$\text{minmax} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad \text{and} \tag{19}$$

$$\text{standard anomaly} = \frac{x - \mu}{\sigma}, \tag{20}$$

respectively. In Eq. (19), $x_{\min}$ is the minimum value within the training dataset for some input feature x while $x_{\max}$ is the maximum value in the training dataset. In Eq. (20), μ is the mean of feature x in the training dataset and σ is the standard deviation. For this paper, the standard anomaly is used.

Using all available input features yields an accuracy of 90%, 84%, 86%, 91%, 90%, and 89% for logistic regression, naïve Bayes, decision tree, random forest, gradient boosted trees, and support vector machines, respectively. Beyond the relatively good accuracy, the ROC curves are shown in Fig. 10a. This time there are generally two sets of curves, one better performing group (logistic regression, random forest, gradient boosted trees, and support vector machines) with AUCs of 0.97 and a worse performing group (naïve Bayes and decision tree) AUCs around 0.87. This separation coincides with the flexibility of the classification methods. The better performing groups are better set to deal with many features and nonlinear interactions of the features, while the worse performing group is a bit more restricted in how it combines many features. Considering the performance diagram (Fig. 10b), the same grouping of high AUC performing models have higher CSI scores (>0.8) and have little to no frequency bias. Meanwhile the lower AUC performing models have lower CSI (0.75) and NB has a slight overforecasting bias. Overall, the ML performance on classifying if an image has a thunderstorm is doing well with all predictors. While a good performing model is a desired outcome of ML, at this point we do not know how the ML is making its predictions. This is part of the "black-box" issue of ML and does not lend itself to being consistent with the ML user's prior knowledge (see note in introduction on consistency; Murphy 1993).

To alleviate some of opaqueness of the ML black box, one can interrogate the trained ML models by asking: "What input features are most important to the decision?" and "Are the patterns the ML models learned physical (e.g., follow meteorological expectation)?" The techniques named permutation importance (Breiman 2001; Lakshmanan et al. 2015) and accumulated local effects (ALE; Apley and Zhu 2020) are used to answer these two questions, respectively. Permutation importance is a method in which the relative importance of an input feature is quantified by considering the change in

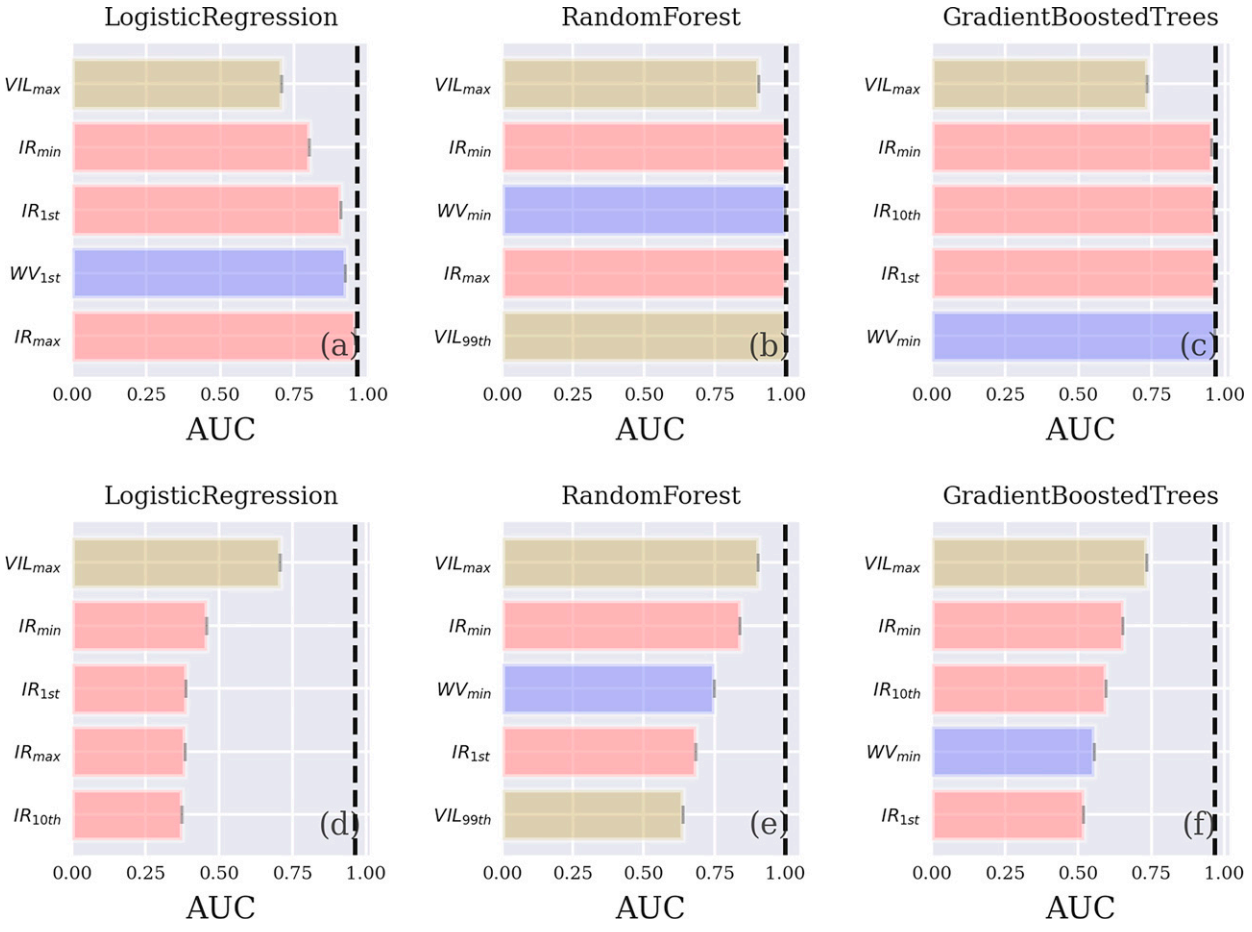

FIG. 11. Backward permutation importance test for the best performing classification ML models. Single pass results are in the top row, while multi-pass forward results are for the bottom row. Each column corresponds to a different ML method: (a),(d) logistic regression; (b),(e) random forest; and (c),(f) gradient boosted trees. Bars are colored by their source, yellow for the vertically integrated liquid (VIL), red for the infrared (IR), blue for water vapor (WV), and black for visible (VIS). Number subscripts correspond to the percentile of that variable. The dashed black line is the original AUC value when all features are not shuffled.

evaluation metric (e.g., AUC) when that input variable is shuffled (i.e., randomized). The intuition is that the most important variables when shuffled will cause the largest change to the evaluation metric. There are two main flavors of permutation importance, named single-pass and multi-pass. Single-pass permutation importance goes through each input variable and shuffles them one by one, calculating the change in the evaluation metrics. Multi-pass permutation importance uses the result of the single-pass, but progressively permutes features. In other words, features are successively permuted in the order that they were determined as important (most important then second most important etc.) from the single pass but are now left shuffled. The specific name for the method we have been describing is the *backward multi-pass permutation importance*. The backward name comes from the direction of shuffling, starting will all variables unshuffled and shuffling more and more of them. There is the opposite direction, named *forward multi-pass permutation importance*, where the starting point is that all features are shuffled to start. Then each feature is unshuffled in order of their importance from the single-pass permutation importance. For visual learners, see the animations (for the backward direction; Figs. ES4 and ES5) in the supplement of McGovern et al. (2019). The reason for doing multi-pass permutation importance is because correlated features could result in falsely identifying unimportant variables using the single pass permutation importance. The best analysis of the permutation test is to use both the single pass and multi-pass tests in conjunction.

The top five most important features for the better performing models (i.e., logistic regression, random forest, and gradient boosted trees) as determined by permutation importance are shown in Fig. 11. For all ML methods both the single and multi-pass test show that the maximum vertically integrated liquid is the most important feature, while the minimum brightness temperature from the clean infrared and midtropospheric water vapor channels are found within the top five predictors (except multi-pass test for logistic regression). In general, the way to interpret these is to take the consensus over all models which features are important. At this point it time to consider if the most important predictors

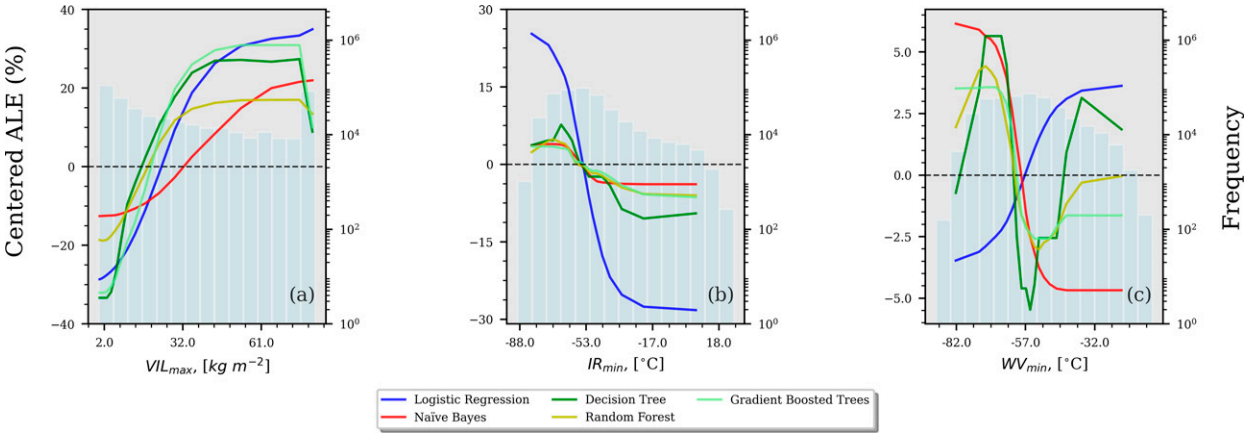

FIG. 12. Accumulated local effects (ALE) for (a) the maximum vertically integrated liquid (VIL_{max}), (b) the minimum brightness temperature from infrared (IR_{min}), and (c) the minimum brightness temperature from the water vapor channel (WV_{min}). Lines correspond to all the ML methods trained (except support vector machines) and colors match Fig. 9. Gray histograms in the background are the counts of points in each bin.

make meteorological sense. Vertically integrated liquid has been shown to have a relationship to lightning (e.g., Watson et al. 1995) and is thus plausible to be the most important predictor. Similarly, the minimum brightness temperature at the water vapor and clean infrared channels also makes physical sense because lower temperatures are generally associated with taller storms. We could also reconcile the maximum infrared brightness temperature (Fig. 11a) as a proxy for the surface temperature which correlates to buoyancy, but note that the relative change in AUC with this feature is quite small. Conversely, any important predictors that do not align with traditional meteorological knowledge may require further exploration to determine why the model is placing such weight on those variables. Does the predictor have some statistical correlation with the meteorological event that is unexplained by past literature, or are there nonphysical characteristics of the data that may be influencing the model during training? In the latter case, it is possible that your model might be getting the right answer for the wrong reasons.

Meanwhile minimum brightness temperature at both the water vapor and clean infrared channels also makes physical sense since lower temperatures are related with taller storms. We could also reconcile the max infrared brightness temperature as a proxy for the surface temperature, which correlates to buoyancy, but not that the relative change in AUC with this feature is quite small. If any the top predictors do not make sense meteorologically, then your model might be getting the right answer for the wrong reasons.

Accumulated local effects are where small changes to input features and their associated change on the output of the model are quantified. The goal behind ALE is to investigate the relationship between an input feature and the output. ALE is performed by binning the data based on the feature of interest. Then for each example in each bin, the feature value is replaced by the edges of the bin. The mean difference in the model output from the replaced feature value is then used as the ALE for that bin. This process is repeated for all bins which result in a curve. For example, the ALE for some of the top predictors of the permutation test is shown in in Fig. 12. At this step, the ALEs can be mainly used to see if the ML models have learned physically plausible trends with input features. For the vertically integrated liquid, all models show that as the max vertically integrated liquid increases from about 2 to 30 kg m^{-2} the average output probability of the model will increase, but values larger than 30 kg m^{-2} generally all have the same local effect on the prediction (Fig. 12a). As for the minimum clean infrared brightness temperature, the magnitude of the average change is considerably different across the different models, but generally all have the same pattern. As the minimum temperature increases from −88° to −55°C, the mean output probability decreases: temperatures larger than −17°C have no change (Fig. 12b). Last, all models but the logistic regression show a similar pattern with the minimum water vapor brightness temperature, but notice the magnitude of the y axis (Fig. 12c). Much less change occurs with this feature. For interested readers, additional interpretation techniques and examples can be found in Molnar (2022).

2) REGRESSION

As stated in section 3a, task 2 is to predict the number of lightning flashes inside an image. Thus, the regression methods available to do this task are linear regression, decision tree, random forest, gradient boosted trees, and support vector machines. Similar to task 1 a simple scenario is considered first, using T_b as the lone predictor. Figure 13 shows the general relationship between T_b and the number of flashes in the image. For $T_b > -25$°C, most images do not have any lightning, while $T_b < -25$°C shows a general increase of lightning flashes. Given there are a lot of images with zero flashes (approximately 50% of the total dataset; black points in Fig. 13), the linear methods will likely struggle to capture a skillful prediction. One way to improve performance would be to only

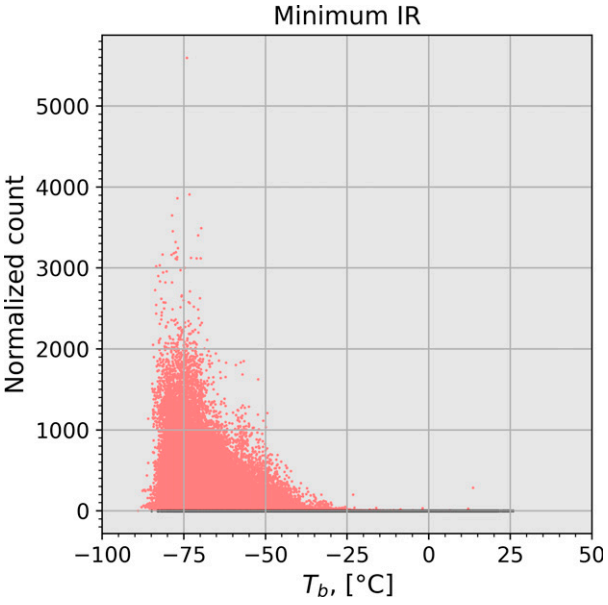

FIG. 13. The training data relationship between the minimum brightness temperature from infrared (T_b) and the number of flashes detected by GLM. All non-thunderstorm images (number of flashes equal to 0) are in black.

predict the number of flashes on images where there are nonzero flashes. While this might not seem like a viable way forward since non-lightning cases would be useful to predict, in practice we could leverage the very good performance of the classification model from section 3c(1), and then use the trained regression on images that are confident to have at least one flash in them. An example of this done in the literature is Gagne et al. (2017), where hail size predictions were only made if the classification model said there was hail.

As before, all methods are fit on the training data initially using the default hyper-parameters. A common way to compare regression model performance is to create a one-to-one plot, which has the predicted number of flashes on the x axis and the true measured number of flashes on the y axis. A perfect model will show all points tightly centered along the diagonal of the plot. This is often the quickest qualitative assessment of how a regression model is performing. While T_b was well suited for the classification of thunderstorm/no-thunderstorm, it is clear that fitting a linear model to the data in Fig. 13 did not do well (Figs. 14a,e), leading to a strong overprediction of the number of lightning flashes in an images with less than 100 flashes, while under-predicting the number of flashes for images with more than 100 flashes. The tree based methods tend to do better, but there is still a large amount of scatter and an over estimation of storms with less than 100 flashes.

To tie quantitative metrics to the performance of each model the following are common metrics calculated: mean bias, mean absolute error (MAE), root mean squared error (RMSE) and coefficient of determination (R^2). Their mathematical representations are the following:

$$\text{bias} = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j), \tag{21}$$

$$\text{MAE} = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|, \tag{22}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}, \tag{23}$$

$$R^2 = 1 - \frac{\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{N}(y_j - \bar{y})^2}. \tag{24}$$

All of these metrics are shown in Fig. 15. In general, the metrics give a more quantitative perspective to the one-to-one plots. The poor performance of the linear methods shows, with the two worst performances being the support vector machines and linear regression with biases of 71 and 6 flashes, respectively. While no method provides remarkable performance, the random forest and gradient boosted trees perform better with this single feature model (show better metrics holistically).

As before, the next logical step is to use all available features to predict the number of flashes: those results are found in Figs. 16 and 17. As expected, the model performance increases. Now all models show a general correspondence between the predicted number of flashes and the true number of flashes in the one-to-one plot (Fig. 16). Meanwhile the scatter for random forest and gradient boosted trees has reduced considerably when comparing to the single input models (Figs. 16c,d). While comparing the bias of the models trained with all predictors is relatively similar, the other metrics are much improved, showing large reductions in MAE and RMSE and increases in R^2 (Fig. 17) for all methods except decision trees. This reinforces that fact that similar to the classification example, it is always good to compare more than one metric.

Since the initial fitting of the ML models used the default parameters, there might be room for tuning the models to have better performance. Here we will show an example of some hyper-parameter tuning of a random forest. The common parameters that can be altered in a random forest include the following: the maximum depth of the trees (i.e., number of decisions in a tree) and the number of trees in the forest. The formal hyper-parameter search will use the full training dataset, and systematically vary the depth of the trees from 1 to 10 (in increments of 1) as well as the number of trees from 1 to 100 (1, 5, 10, 25, 50, 100). This results in 60 total models that are trained.

To evaluate which is the best configuration, the same metrics as before are shown in Fig. 18 as a function of the depth of the trees. The random forest quickly gains skill with added depth beyond one, with all metrics improving for both the training (dashed lines) and validation datasets (solid lines). Beyond a depth of four, the bias, MAE, and RMSE all stagnate, but the

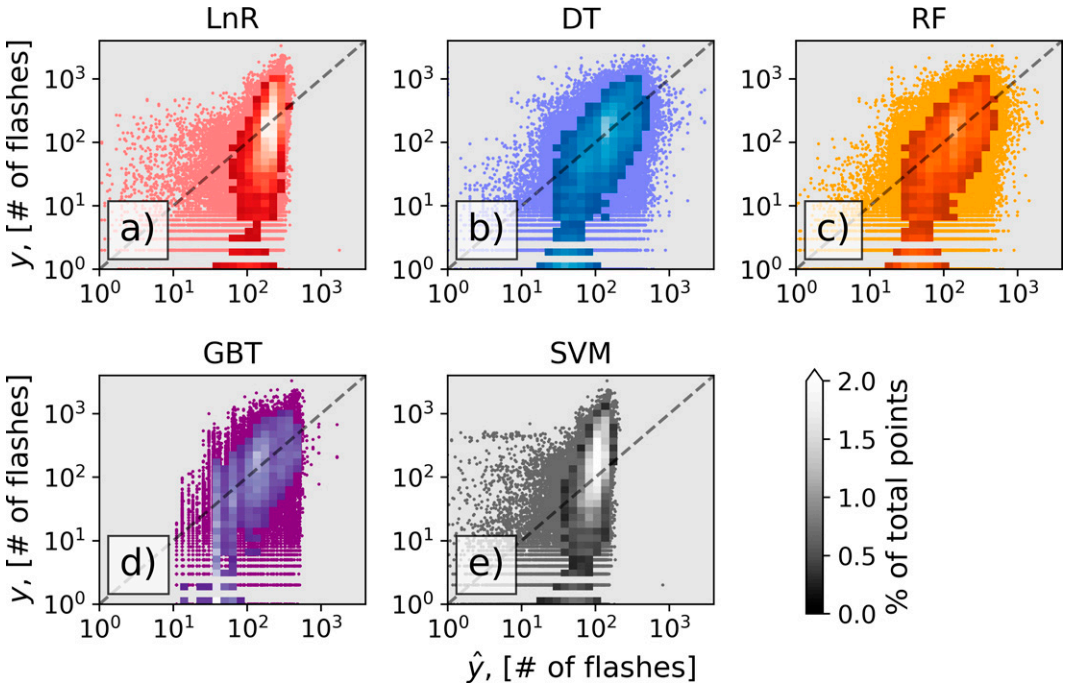

FIG. 14. The one-to-one relationship between the predicted number of lightning flashes from the ML learning models trained on only T_b (x axis; $\hat{y}$) and the number of measure flashes from GLM (y axis; y). Each marker is one observation. Meanwhile areas with more than 100 points in close proximity are shown in the colored boxes. The lighter the shade of the color, the higher the density of points. (a) Linear regression (LnR; reds), (b) decision tree (DT; blues), (c) random forest (RF; oranges), (d) gradient boosted trees (GBT; purples), and (e) linear support vector machines (SVM; grays).

R^2 value increases until a depth of eight where the training data continue to increase. There does not seem to be that large of an effect of increasing the number of trees beyond 10 (color change of lines). The characteristic of increasing training metric skills but no increase (or a decrease) to validation data skill is the overfitting signal we discussed in section 3b. Thus, the best random forest model choice for predicting lightning flashes is a random forest with a max depth of eight and a total of 10 trees. The reason we choose 10 trees, is because in general choosing a simpler model is less computationally expensive to use as well as a more interpretable than a model with 1000 trees.

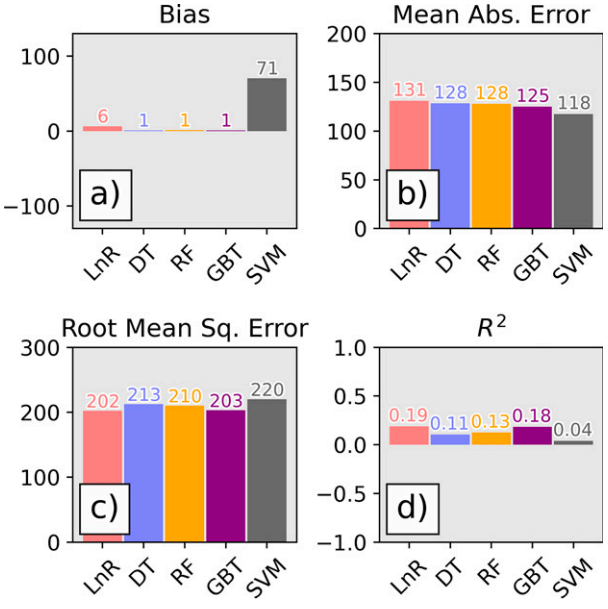

FIG. 15. Validation dataset metrics for all ML models. Colors are as in Fig. 14. Exact numerical value is reported on top of each bar.

d. *Testing*

As mentioned before, the test dataset is the dataset you hold out until the end when all hyper-parameter tuning has finished so that there is no unintentional tuning of the final model configuration to a dataset. Thus, now that we have evaluated the performance of all our models on the validation dataset it is time to run the same evaluations as in sections 3c(1) and 3c(2). These test results are the end performance metrics that should be interpreted as the expected ML performance on new data (e.g., the ML applied in practice). For the ML models here, the metrics are very similar as the validation set. (For brevity the extra figures are included in the appendix Figs. A1–A3.)

4. Summary and future work

This manuscript was the first of two machine learning (ML) tutorial papers designed for the operational meteorology community. This paper supplied a survey of some of the most

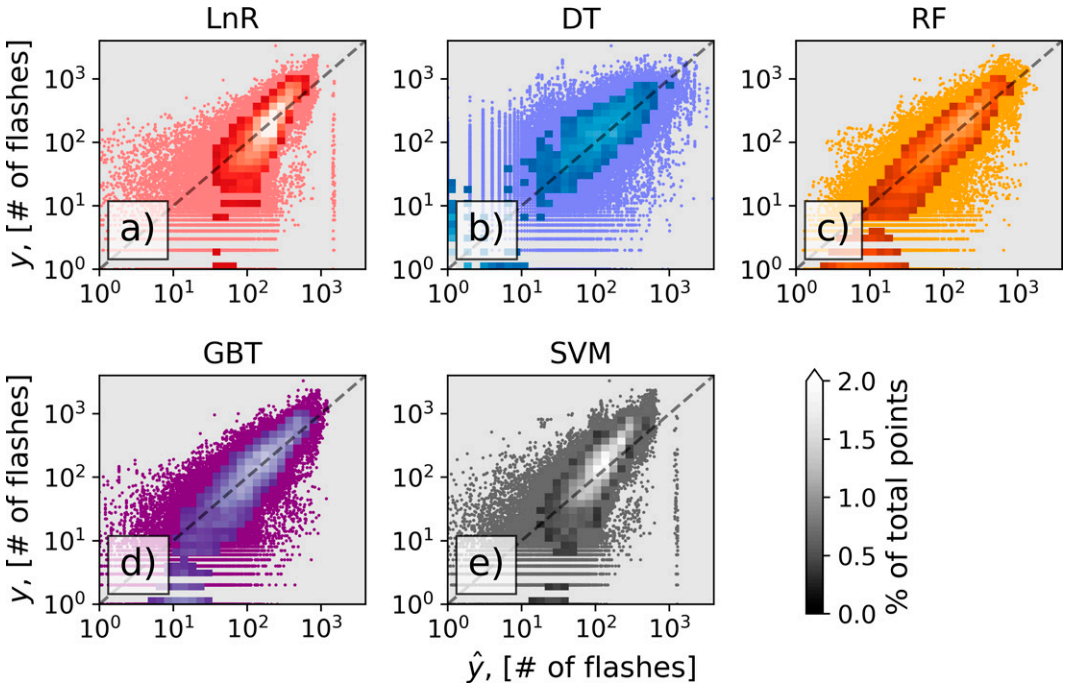

FIG. 16. As in Fig. 14, but now the x axis is provided from the ML models trained with all available input features.

common ML methods. All ML methods described here are considered *supervised* methods, meaning the data the models are trained from include pre-labeled *truth* data. The specific methods covered included linear regression, logistic regression, decision trees, random forests, gradient boosted decision trees, naïve Bayes, and support vector machines. The overarching goal of the paper was to introduce the ML methods in such a way that ML methods are more familiar to readers as they encounter them in the operational community and within the general meteorological literature. Moreover, this manuscript provided ample references of published meteorological examples as well as open-source code to act as catalysts for readers to adapt and try ML on their own datasets and in their workflows.

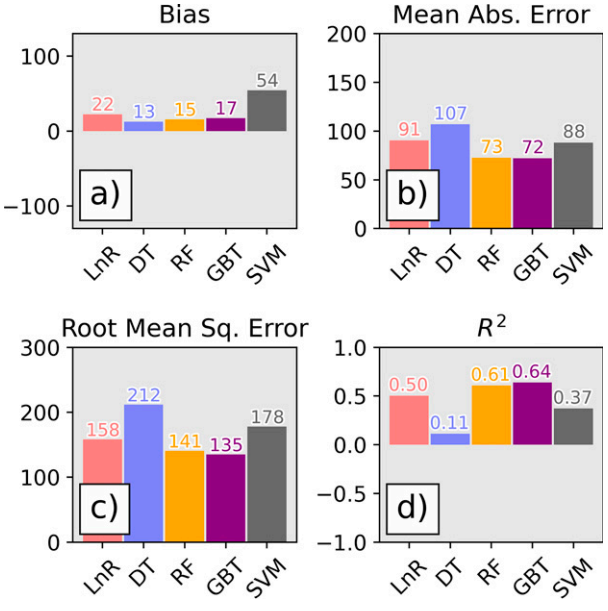

FIG. 17. As in Fig. 15, but for ML models trained with all available input features.

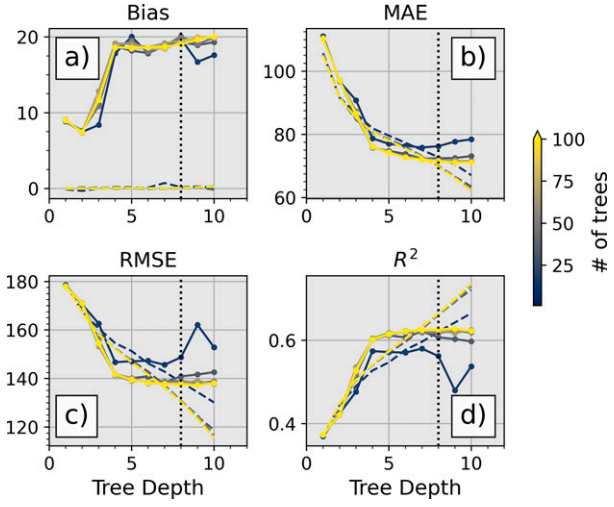

FIG. 18. Hyper-parameter tuning of a random forest for predicting the number of lightning flashes. All input features are used. Solid lines are the validation dataset while the dashed lines are the training data. The vertical dotted line is the depth of trees where overfitting begins.

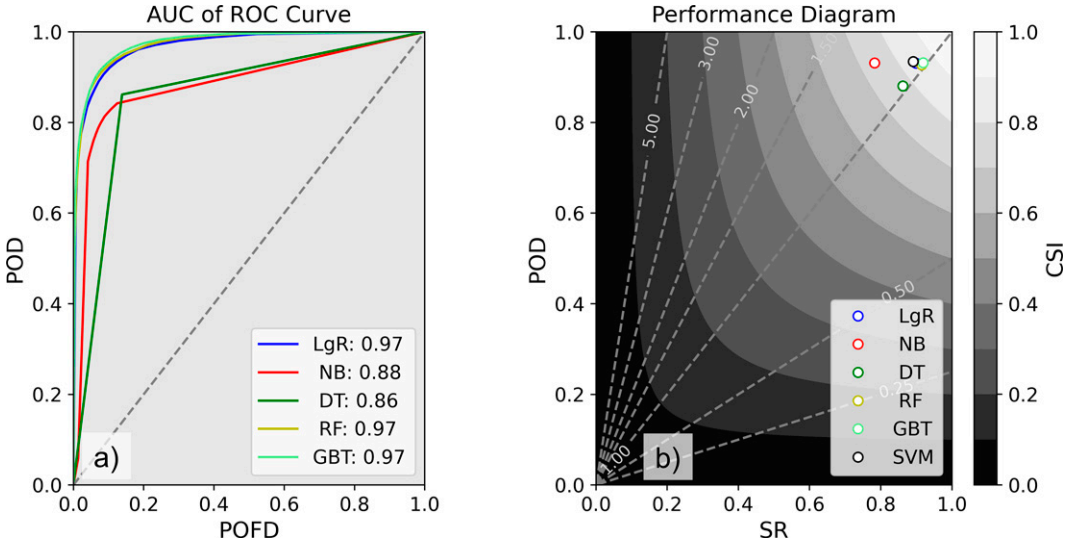

FIG. A1. As in Fig. 9, but now for the test dataset.

Additionally, this manuscript provided a tutorial example of how to apply ML to a couple meteorological tasks using the Storm Event Imagery dataset (SEVIR; Veillette et al. 2020) dataset. We

1) Discussed the various steps of preparing data for ML (i.e., removing artifacts; engineering features, training/validation/testing splits; section 3b).
2) Conducted a classification task to predict if satellite images had lightning within them. This section included discussions of training, evaluation and interrogation of the trained ML models [section 3c(1)].
3) Exhibited a regression task to predict the number of lightning flashes in a satellite image. This section also contained discussions of training/evaluation as well as an example of hyper-parameter tuning [section 3c(2)].
4) Released python code to conduct all steps and examples in this manuscript (see data availability statement).

The follow on paper in this series will discuss a more complex, yet potentially more powerful, grouping of ML methods: neural networks and deep learning. Like a lot of the ML methods described in this paper, neural networks are not necessarily

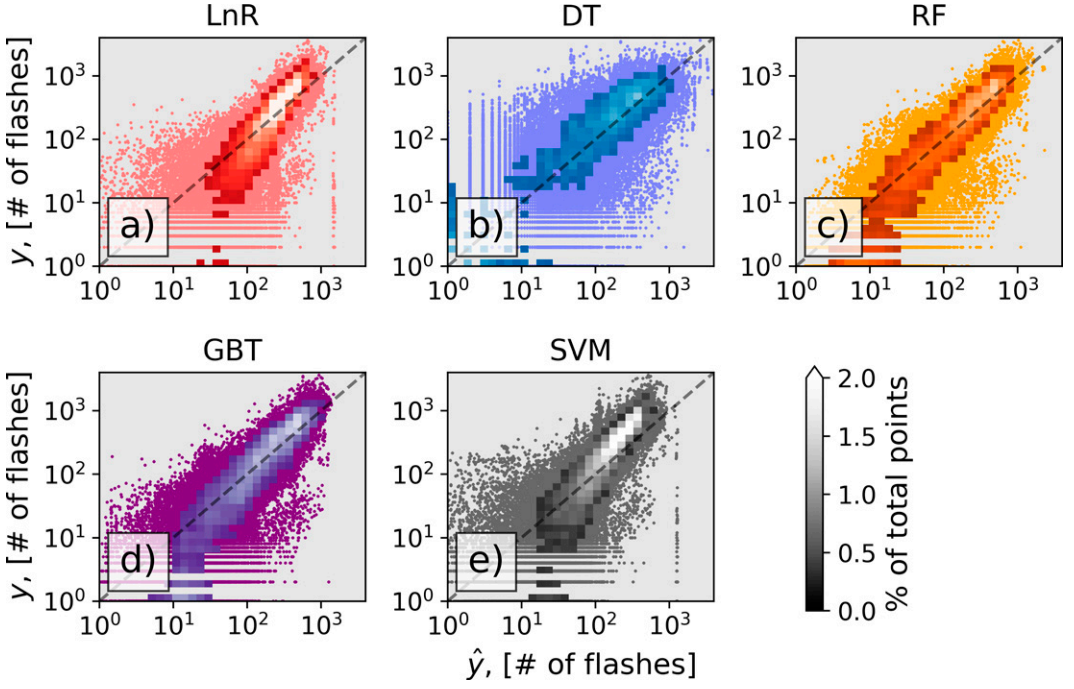

FIG. A2. As in Fig. 14, but for the test dataset.

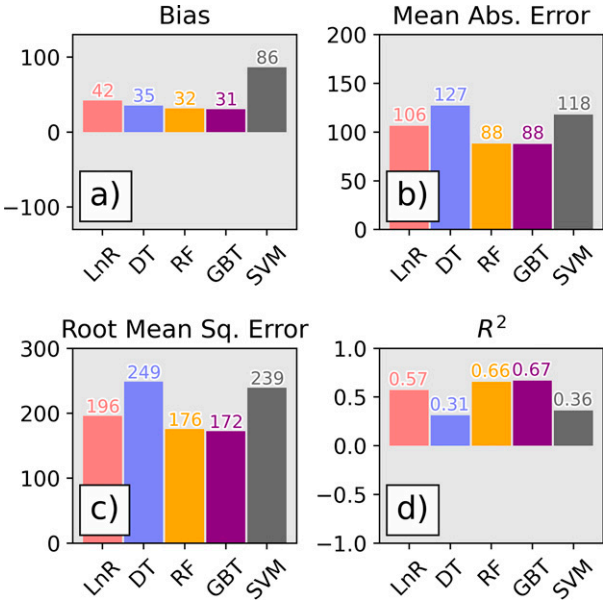

FIG. A3. As in Fig. 15, but for the test dataset.

new (Rumelhart et al. 1986) and were first applied to meteorology topics decades ago (e.g., Key et al. 1989; Lee et al. 1990). Although, given the exponential growth of computing resources and dataset sizes, research using neural networks and deep learning in meteorology has been accelerating (e.g., Fig. 1c; Gagne et al. 2019; Lagerquist et al. 2020; Cintineo et al. 2020; Chase et al. 2021; Hilburn et al. 2021; Lagerquist et al. 2021; Molina et al. 2021; Ravuri et al. 2021). Thus, it is important that operational meteorologists also understand the basics of neural networks and deep learning.

Acknowledgments. We would like to acknowledge and thank the three anonymous reviewers who provided valuable feedback to this manuscript. This material is based upon work supported by the National Science Foundation under Grant ICER-2019758, supporting authors RJC, AM, and AB. Author DRH was provided support by NOAA/Office of Oceanic and Atmospheric Research under NOAA–University of Oklahoma Cooperative Agreements NA16OAR4320115 and NA21OAR4320204, U.S. Department of Commerce. The scientific results and conclusions, as well as any views or opinions expressed herein, are those of the authors and do not necessarily reflect the views of NOAA or the Department of Commerce. We want to acknowledge the work put forth by the authors of the SEVIR dataset (Mark S. Veillette, Siddharth Samsi, and Christopher J. Mattioli) for making a high-quality free dataset. We would also like to acknowledge the open-source python community for providing their tools for free. Specifically, we acknowledge Google Colab (Bisong 2019), Anaconda (Anaconda 2020), scikit-learn (Pedregosa et al. 2011), Pandas (Wes McKinney 2010), Numpy (Harris et al. 2020), and Jupyter (Kluyver et al. 2016).

Data availability statement. As an effort to catalyze the use and trust of machine learning within meteorology we have supplied a github repository with a code tutorial of a lot of the same things discussed in this paper. The latest version of github repository can be located here: https://github.com/ai2es/WAF_ML_Tutorial_Part1. If you are interested in the version of the repository that was available at time of publication please see the zendo archive of version 1 here: https://zenodo.org/record/6941510. The original github repo for SEVIR is located here: https://github.com/MIT-AI-Accelerator/neurips-2020-sevir.

APPENDIX

Testing Dataset Figures

This appendix contains the test dataset evaluations for both the classification task (Fig. A1) and the regression task (Figs. A2 and A3). Results are largely the same as the validation set, so to save space they were included here.

REFERENCES

Anaconda, 2020: Anaconda software distribution. Anaconda Inc., accessed 1 July 2022, https://docs.anaconda.com/.

Apley, D. W., and J. Zhu, 2020: Visualizing the effects of predictor variables in black box supervised learning models. *J. Roy. Stat. Soc.*, **82B**, 1059–1086, https://doi.org/10.1111/rssb.12377.

Bieli, M., A. H. Sobel, S. J. Camargo, and M. K. Tippett, 2020: A statistical model to predict the extratropical transition of tropical cyclones. *Wea. Forecasting*, **35**, 451–466, https://doi.org/10.1175/WAF-D-19-0045.1.

Billet, J., M. DeLisi, B. Smith, and C. Gates, 1997: Use of regression techniques to predict hail size and the probability of large hail. *Wea. Forecasting*, **12**, 154–164, https://doi.org/10.1175/1520-0434(1997)012<0154:UORTTP>2.0.CO;2.

Bischl, B., O. Mersmann, H. Trautmann, and C. Weihs, 2012: Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evol. Comput.*, **20**, 249–275, https://doi.org/10.1162/EVCO_a_00069.

Bisong, E., Ed., 2019: Google colaboratory. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Apress, 59–64, https://doi.org/10.1007/978-1-4842-4470-8_7.

Bonavita, M., and Coauthors, 2021: Machine learning for earth system observation and prediction. *Bull. Amer. Meteor. Soc.*, **102**, E710–E716, https://doi.org/10.1175/BAMS-D-20-0307.1.

Breiman, L., 1984: *Classification and Regression Trees.* Routledge, 368 pp.

——, 2001: Random forests. *Mach. Learn.*, **45**, 5–32, https://doi.org/10.1023/A:1010933404324.

Burke, A., N. Snook, D. J. Gagne II, S. McCorkle, and A. McGovern, 2020: Calibration of machine learning–based probabilistic hail predictions for operational forecasting. *Wea. Forecasting*, **35**, 149–168, https://doi.org/10.1175/WAF-D-19-0105.1.

Cains, M. G., and Coauthors, 2022: NWS forecasters' perceptions and potential uses of trustworthy AI/ML for hazardous weather risks. *21st Conf. on Artificial Intelligence for Environmental Science*, Houston, TX, Amer. Meteor. Soc., 1.3, https://ams.confex.com/ams/102ANNUAL/meetingapp.cgi/Paper/393121.

Chase, R. J., S. W. Nesbitt, and G. M. McFarquhar, 2021: A dual-frequency radar retrieval of two parameters of the snowfall particle size distribution using a neural network. *J. Appl.*

Meteor. Climatol., **60**, 341–359, https://doi.org/10.1175/JAMC-D-20-0177.1.

Chisholm, D., J. Ball, K. Veigas, and P. Luty, 1968: The diagnosis of upper-level humidity. *J. Appl. Meteor.*, **7**, 613–619, https://doi.org/10.1175/1520-0450(1968)007<0613:TDOULH>2.0.CO;2.

Cintineo, J. L., M. Pavolonis, J. Sieglaff, and D. Lindsey, 2014: An empirical model for assessing the severe weather potential of developing convection. *Wea. Forecasting*, **29**, 639–653, https://doi.org/10.1175/WAF-D-13-00113.1.

——, and Coauthors, 2018: The NOAA/CIMSS ProbSevere model: Incorporation of total lightning and validation. *Wea. Forecasting*, **33**, 331–345, https://doi.org/10.1175/WAF-D-17-0099.1.

——, M. J. Pavolonis, J. M. Sieglaff, L. Cronce, and J. Brunner, 2020: NOAA Probsevere v2.0—Probhail, Probwind, and Probtor. *Wea. Forecasting*, **35**, 1523–1543, https://doi.org/10.1175/WAF-D-19-0242.1.

Conrick, R., J. P. Zagrodnik, and C. F. Mass, 2020: Dual-polarization radar retrievals of coastal Pacific Northwest raindrop size distribution parameters using random forest regression. *J. Atmos. Oceanic Technol.*, **37**, 229–242, https://doi.org/10.1175/JTECH-D-19-0107.1.

Cui, W., X. Dong, B. Xi, and Z. Feng, 2021: Climatology of linear mesoscale convective system morphology in the United States based on the random-forests method. *J. Climate*, **34**, 7257–7276, https://doi.org/10.1175/JCLI-D-20-0862.1.

Czernecki, B., M. Taszarek, M. Marosz, M. Półrolniczak, L. Kolendowicz, A. Wyszogrodzki, and J. Szturc, 2019: Application of machine learning to large hail prediction—The importance of radar reflectivity, lightning occurrence and convective parameters derived from ERA5. *Atmos. Res.*, **227**, 249–262, https://doi.org/10.1016/j.atmosres.2019.05.010.

Elmore, K. L., and H. Grams, 2016: Using mPING data to generate random forests for precipitation type forecasts. *14th Conf. on Artificial and Computational Intelligence and its Applications to the Environmental Sciences*, New Orleans, LA, Amer. Meteor. Soc., 4.2, https://ams.confex.com/ams/96Annual/webprogram/Paper289684.html.

Flora, M. L., C. K. Potvin, P. S. Skinner, S. Handler, and A. McGovern, 2021: Using machine learning to generate storm-scale probabilistic guidance of severe weather hazards in the Warn-on-Forecast system. *Mon. Wea. Rev.*, **149**, 1535–1557, https://doi.org/10.1175/MWR-D-20-0194.1.

Friedman, J., 2001: Greedy function approximation: A gradient boosting machine. *Ann. Stat.*, **29**, 1189–1232, https://doi.org/10.1214/aos/1013203451.

Gagne, D., A. McGovern, and J. Brotzge, 2009: Classification of convective areas using decision trees. *J. Atmos. Oceanic Technol.*, **26**, 1341–1353, https://doi.org/10.1175/2008JTECHA1205.1.

——, ——, ——, and M. Xue, 2013: Severe hail prediction within a spatiotemporal relational data mining framework. *13th Int. Conf. on Data Mining*, Dallas, TX, Institute of Electrical and Electronics Engineers, 994–1001, https://doi.org/10.1109/ICDMW.2013.121.

——, ——, S. Haupt, R. Sobash, J. Williams, and M. Xue, 2017: Storm-based probabilistic hail forecasting with machine learning applied to convection-allowing ensembles. *Wea. Forecasting*, **32**, 1819–1840, https://doi.org/10.1175/WAF-D-17-0010.1.

——, H. Christensen, A. Subramanian, and A. Monahan, 2019: Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz '96 model. *J. Adv. Model. Earth Syst.*, **12**, e2019MS001896, https://doi.org/10.1029/2019MS001896.

Gensini, V. A., C. Converse, W. S. Ashley, and M. Taszarek, 2021: Machine learning classification of significant tornadoes and hail in the United States using ERA5 proximity soundings. *Wea. Forecasting*, **36**, 2143–2160, https://doi.org/10.1175/WAF-D-21-0056.1.

Glahn, H. R., and D. A. Lowry, 1972: The use of Model Output Statistics (MOS) in objective weather forecasting. *J. Appl. Meteor.*, **11**, 1203–1211, https://doi.org/10.1175/1520-0450(1972)011<1203:TUOMOS>2.0.CO;2.

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep Learning*. MIT Press, 800 pp., http://www.deeplearningbook.org.

Grams, H. M., P.-E. Kirstetter, and J. J. Gourley, 2016: Naïve Bayesian precipitation type retrieval from satellite using a cloud-top and ground-radar matched climatology. *J. Hydrometeor.*, **17**, 2649–2665, https://doi.org/10.1175/JHM-D-16-0058.1.

Harris, C. R., and Coauthors, 2020: Array programming with NumPy. *Nature*, **585**, 357–362, https://doi.org/10.1038/s41586-020-2649-2.

Herman, G., and R. Schumacher, 2018a: Dendrology in numerical weather prediction: What random forests and logistic regression tell us about forecasting. *Mon. Wea. Rev.*, **146**, 1785–1812, https://doi.org/10.1175/MWR-D-17-0307.1.

——, and ——, 2018b: Money doesn't grow on trees, but forecasts do: Forecasting extreme precipitation with random forests. *Mon. Wea. Rev.*, **146**, 1571–1600, https://doi.org/10.1175/MWR-D-17-0250.1.

Hilburn, K. A., I. Ebert-Uphoff, and S. D. Miller, 2021: Development and interpretation of a neural-network-based synthetic radar reflectivity estimator using GOES-R satellite observations. *J. Appl. Meteor. Climatol.*, **60**, 3–21, https://doi.org/10.1175/JAMC-D-20-0084.1.

Hill, A. J., and R. S. Schumacher, 2021: Forecasting excessive rainfall with random forests and a deterministic convection-allowing model. *Wea. Forecasting*, **36**, 1693–1711, https://doi.org/10.1175/WAF-D-21-0026.1.

——, G. R. Herman, and R. S. Schumacher, 2020: Forecasting severe weather with random forests. *Mon. Wea. Rev.*, **148**, 2135–2161, https://doi.org/10.1175/MWR-D-19-0344.1.

Hoerl, A. E., and R. W. Kennard, 1970: Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, **12**, 55–67, https://doi.org/10.1080/00401706.1970.10488634.

Holte, R. C., 1993: Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, **11**, 63–90, https://doi.org/10.1023/A:1022631118932.

Hu, L., E. A. Ritchie, and J. S. Tyo, 2020: Short-term tropical cyclone intensity forecasting from satellite imagery based on the deviation angle variance technique. *Wea. Forecasting*, **35**, 285–298, https://doi.org/10.1175/WAF-D-19-0102.1.

Hubbert, J. C., M. Dixon, S. M. Ellis, and G. Meymaris, 2009: Weather radar ground clutter. Part I: Identification, modeling, and simulation. *J. Atmos. Oceanic Technol.*, **26**, 1165–1180, https://doi.org/10.1175/2009JTECHA1159.1.

Jergensen, G. E., A. McGovern, R. Lagerquist, and T. Smith, 2020: Classifying convective storms using machine learning. *Wea. Forecasting*, **35**, 537–559, https://doi.org/10.1175/WAF-D-19-0170.1.

Kalnay, E., 2002: *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 341 pp., https://doi.org/10.1017/CBO9780511802270.

Key, J., J. Maslanik, and A. Schweiger, 1989: Classification of merged AVHRR and SMMR Arctic data with neural networks. *Photogramm. Eng. Remote Sens.*, **55**, 1331–1338.

Kluyver, T., and Coauthors, 2016: Jupyter Notebooks—A publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds., IOS Press, 87–90.

Kossin, J. P., and M. Sitkowski, 2009: An objective model for identifying secondary eyewall formation in hurricanes. *Mon. Wea. Rev.*, **137**, 876–892, https://doi.org/10.1175/2008MWR2701.1.

Kühnlein, M., T. Appelhans, B. Thies, and T. Nauß, 2014: Precipitation estimates from MSG SEVIRI daytime, nighttime, and twilight data with random forests. *J. Appl. Meteor. Climatol.*, **53**, 2457–2480, https://doi.org/10.1175/JAMC-D-14-0082.1.

Kuncheva, L. I., 2006: On the optimality of naïve Bayes with dependent binary features. *Pattern Recognit. Lett.*, **27**, 830–837, https://doi.org/10.1016/j.patrec.2005.12.001.

Kurdzo, J. M., E. F. Joback, P.-E. Kirstetter, and J. Y. N. Cho, 2020: Geospatial QPE accuracy dependence on weather radar network configurations. *J. Appl. Meteor. Climatol.*, **59**, 1773–1792, https://doi.org/10.1175/JAMC-D-19-0164.1.

Lackmann, G., Ed., 2011: Numerical weather prediction/data assimilation. *Midlatitude Synoptice Meteorology: Dynamics, Analysis, and Forecasting*, Amer. Meteor. Soc., 274–287.

Lagerquist, R., A. McGovern, and T. Smith, 2017: Machine learning for real-time prediction of damaging straight-line convective wind. *Wea. Forecasting*, **32**, 2175–2193, https://doi.org/10.1175/WAF-D-17-0038.1.

——, ——, C. R. Homeyer, D. J. Gagne II, and T. Smith, 2020: Deep learning on three-dimensional multiscale data for next-hour tornado prediction. *Mon. Wea. Rev.*, **148**, 2837–2861, https://doi.org/10.1175/MWR-D-19-0372.1.

——, J. Q. Stewart, I. Ebert-Uphoff, and C. Kumler, 2021: Using deep learning to nowcast the spatial coverage of convection from *Himawari-8* satellite data. *Mon. Wea. Rev.*, **149**, 3897–3921, https://doi.org/10.1175/MWR-D-21-0096.1.

Lakshmanan, V., C. Karstens, J. Krause, K. Elmore, A. Ryzhkov, and S. Berkseth, 2015: Which polarimetric variables are important for weather/no-weather discrimination? *J. Atmos. Oceanic Technol.*, **32**, 1209–1223, https://doi.org/10.1175/JTECH-D-13-00205.1.

Lee, C.-Y., S. J. Camargo, F. Vitart, A. H. Sobel, J. Camp, S. Wang, M. K. Tippett, and Q. Yang, 2020: Subseasonal predictions of tropical cyclone occurrence and ACE in the S2S dataset. *Wea. Forecasting*, **35**, 921–938, https://doi.org/10.1175/WAF-D-19-0217.1.

Lee, J., R. Weger, S. Sengupta, and R. Welch, 1990: A neural network approach to cloud classification. *IEEE Trans. Geosci. Remote Sens.*, **28**, 846–855, https://doi.org/10.1109/36.58972.

Li, L., and Coauthors, 2020: A causal inference model based on random forests to identify the effect of soil moisture on precipitation. *J. Hydrometeor.*, **21**, 1115–1131, https://doi.org/10.1175/JHM-D-19-0209.1.

Loken, E. D., A. J. Clark, and C. D. Karstens, 2020: Generating probabilistic next-day severe weather forecasts from convection-allowing ensembles using random forests. *Wea. Forecasting*, **35**, 1605–1631, https://doi.org/10.1175/WAF-D-19-0258.1.

——, ——, and A. McGovern, 2022: Comparing and interpreting differently designed random forests for next-day severe weather hazard prediction. *Wea. Forecasting*, **37**, 871–899, https://doi.org/10.1175/WAF-D-21-0138.1.

Malone, T., 1955: Application of statistical methods in weather prediction. *Proc. Natl. Acad. Sci. USA*, **41**, 806–815, https://doi.org/10.1073/pnas.41.11.806.

Mao, Y., and A. Sorteberg, 2020: Improving radar-based precipitation nowcasts with machine learning using an approach based on random forest. *Wea. Forecasting*, **35**, 2461–2478, https://doi.org/10.1175/WAF-D-20-0080.1.

McCorkel, J., J. Van Naarden, D. Lindsey, B. Efremova, M. Coakley, M. Black, and A. Krimchansky, 2019: *GOES-17* advanced baseline imager performance recovery summary. (*IGARSS 2019) 2019 IEEE Int. Geoscience and Remote Sensing Symp.*, Yokohama, Japan, Institute of Electrical and Electronics Engineers, 1–4, https://doi.org/10.1109/IGARSS40859.2019.9044466.

McGovern, A., D. Gagne, J. Williams, R. Brown, and J. Basara, 2014: Enhancing understanding and improving prediction of severe weather through spatiotemporal relational learning. *Mach. Learn.*, **95**, 27–50, https://doi.org/10.1007/s10994-013-5343-x.

——, ——, J. Basara, T. Hamill, and D. Margolin, 2015: Solar energy prediction: An international contest to initiate interdisciplinary research on compelling meteorological problems. *Bull. Amer. Meteor. Soc.*, **96**, 1388–1395, https://doi.org/10.1175/BAMS-D-14-00006.1.

——, R. Lagerquist, D. Gagne, G. Jergensen, K. Elmore, C. Homeyer, and T. Smith, 2019: Making the black box more transparent: Understanding the physical implications of machine learning. *Bull. Amer. Meteor. Soc.*, **100**, 2175–2199, https://doi.org/10.1175/BAMS-D-18-0195.1.

——, I. Ebert-Uphoff, D. J. Gagne II, and A. Bostrom, 2021: The need for ethical, responsible, and trustworthy artificial intelligence for environmental sciences. arXiv, 2112.08453, https://arxiv.org/abs/2112.08453.

McKinney, W., 2010: Data structures for statistical computing in Python. *Proceedings of the Ninth Python in Science Conference*, S. van der Walt and J. Millman, Eds., 56–61, https://doi.org/10.25080/Majora-92bf1922-00a.

Mecikalski, J., J. Williams, C. Jewett, D. Ahijevych, A. LeRoy, and J. Walker, 2015: Probabilistic 0–1-h convective initiation nowcasts that combine geostationary satellite observations and numerical weather prediction model data. *J. Appl. Meteor. Climatol.*, **54**, 1039–1059, https://doi.org/10.1175/JAMC-D-14-0129.1.

Molina, M. J., D. J. Gagne, and A. F. Prein, 2021: A benchmark to test generalization capabilities of deep learning methods to classify severe convective storms in a changing climate. *Earth Space Sci.*, **8**, e2020EA001490, https://doi.org/10.1029/2020EA001490.

Molnar, C., 2022: *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* 2nd ed. 329 pp., https://christophm.github.io/interpretable-ml-book.

Muñoz-Esparza, D., R. D. Sharman, and W. Deierling, 2020: Aviation turbulence forecasting at upper levels with machine learning techniques based on regression trees. *J. Appl. Meteor. Climatol.*, **59**, 1883–1899, https://doi.org/10.1175/JAMC-D-20-0116.1.

Murphy, A. H., 1993: What is a good forecast? An essay on the nature of goodness in weather forecasting. *Wea. Forecasting*, **8**, 281–293, https://doi.org/10.1175/1520-0434(1993)008<0281:WIAGFA>2.0.CO;2.

Neetu, S., M. Lengaigne, J. Vialard, M. Mangeas, C. Menkes, I. Suresh, J. Leloup, and J. Knaff, 2020: Quantifying the benefits of nonlinear methods for global statistical hindcasts of tropical cyclones intensity. *Wea. Forecasting*, **35**, 807–820, https://doi.org/10.1175/WAF-D-19-0163.1.

Nowotarski, C. J., and A. A. Jensen, 2013: Classifying proximity soundings with self-organizing maps toward improving supercell and tornado forecasting. *Wea. Forecasting*, **28**, 783–801, https://doi.org/10.1175/WAF-D-12-00125.1.

Pedregosa, F., and Coauthors, 2011: Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Peter, J. R., A. Seed, and P. J. Steinle, 2013:Application of a Bayesian classifier of anomalous propagation to single-polarization radar reflectivity data. *J. Atmos. Oceanic Technol.*, **30**, 1985–2005, https://doi.org/10.1175/JTECH-D-12-00082.1.

Quinlan, J., 1993: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 302 pp.

Ravuri, S., and Coauthors, 2021: Skilful precipitation nowcasting using deep generative models of radar. *Nature*, **597**, 672–677, https://doi.org/10.1038/s41586-021-03854-z.

Roebber, P., 2009: Visualizing multiple measures of forecast quality. *Wea. Forecasting*, **24**, 601–608, https://doi.org/10.1175/2008WAF2222159.1.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, 1986: Learning representations by back-propagating errors. *Nature*, **323**, 533–536, https://doi.org/10.1038/323533a0.

Schumacher, R. S., A. J. Hill, M. Klein, J. A. Nelson, M. J. Erickson, S. M. Trojniak, and G. R. Herman, 2021: From random forests to flood forecasts: A research to operations success story. *Bull. Amer. Meteor. Soc.*, **102**, E1742–E1755, https://doi.org/10.1175/BAMS-D-20-0186.1.

Sessa, M. F., and R. J. Trapp, 2020: Observed relationship between tornado intensity and pretornadic mesocyclone characteristics. *Wea. Forecasting*, **35**, 1243–1261, https://doi.org/10.1175/WAF-D-19-0099.1.

Shield, S. A., and A. L. Houston, 2022: Diagnosing supercell environments: A machine learning approach. *Wea. Forecasting*, **37**, 771–785, https://doi.org/10.1175/WAF-D-21-0098.1.

Taillardat, M., A.-L. Fougères, P. Naveau, and O. Mestre, 2019: Forest-based and semiparametric methods for the postprocessing of rainfall ensemble forecasting. *Wea. Forecasting*, **34**, 617–634, https://doi.org/10.1175/WAF-D-18-0149.1.

Tibshirani, R., 1996: Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc.*, **58B**, 267–288, https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Vapnik, V., 1963: Pattern recognition using generalized portrait method. *Autom. Remote Control*, **24**, 774–780.

Veillette, M., S. Samsi, and C. Mattioli, 2020: SEVIR: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *Advances in Neural Information Processing Systems*, H. Larochelle et al., Eds., Vol. 33, Curran Associates, Inc., 22 009–22 019, https://proceedings.neurips.cc/paper/2020/file/fa78a16157fed00d7a80515818432169-Paper.pdf.

Vigaud, N., M. K. Tippett, J. Yuan, A. W. Robertson, and N. Acharya, 2019: Probabilistic skill of subseasonal surface temperature forecasts over North America. *Wea. Forecasting*, **34**, 1789–1806, https://doi.org/10.1175/WAF-D-19-0117.1.

Wang, C., P. Wang, D. Wang, J. Hou, and B. Xue, 2020: Nowcasting multicell short-term intense precipitation using graph models and random forests. *Mon. Wea. Rev.*, **148**, 4453–4466, https://doi.org/10.1175/MWR-D-20-0050.1.

Watson, A. I., R. L. Holle, and R. E. López, 1995: Lightning from two national detection networks related to vertically integrated liquid and echo-top information from WSR-88D radar. *Wea. Forecasting*, **10**, 592–605, https://doi.org/10.1175/1520-0434(1995)010<0592:LFTNDN>2.0.CO;2.

Williams, J., 2014: Using random forests to diagnose aviation turbulence. *Mach. Learn.*, **95**, 51–70, https://doi.org/10.1007/s10994-013-5346-7.

——, D. Ahijevych, S. Dettling, and M. Steiner, 2008a: Combining observations and model data for short-term storm forecasting. *Proc. SPIE*, **7088**, 708805, https://doi.org/10.1117/12.795737.

——, R. Sharman, J. Craig, and G. Blackburn, 2008b: Remote detection and diagnosis of thunderstorm turbulence. *Proc. SPIE*, **7088**, 708804, https://doi.org/10.1117/12.795570.

Yang, L., H. Xu, and S. Yu, 2021: Estimating $PM_{2.5}$ concentrations in contiguous eastern coastal zone of China using MODIS AOD and a two-stage random forest model. *J. Atmos. Oceanic Technol.*, **38**, 2071–2080, https://doi.org/10.1175/JTECH-D-20-0214.1.

Yoshida, S., T. Morimoto, T. Ushio, and Z. Kawasaki, 2009: A fifth-power relationship for lightning activity from Tropical Rainfall Measuring Mission satellite observations. *J. Geophys. Res.*, **114**, D09104, https://doi.org/10.1029/2008JD010370.

Zhang, Z., D. Wang, J. Qiu, J. Zhu, and T. Wang, 2021: Machine learning approaches for improving near-real-time IMERG rainfall estimates by integrating cloud properties from NOAA CDR PATMOS-x. *J. Hydrometeor.*, **22**, 2767–2781, https://doi.org/10.1175/JHM-D-21-0019.1.

Zou, H., and T. Hastie, 2005: Regularization and variable selection via the elastic net. *J. Roy. Stat. Soc.*, **67B**, 301–320, https://doi.org/10.1111/j.1467-9868.2005.00503.x.