UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

USING MACHINE LEARNING TO PREDICT DAMAGING STRAIGHT-LINE CONVECTIVE WINDS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN METEOROLOGY

By

RYAN LAGERQUIST
Norman, Oklahoma
2016

USING MACHINE LEARNING TO PREDICT DAMAGING STRAIGHT-LINE CONVECTIVE WINDS

A THESIS APPROVED FOR THE
SCHOOL OF METEOROLOGY

BY

Dr. Amy McGovern, Chair

Mr. Travis Smith

Dr. Michael Richman

Acknowledgements

This thesis would not have been possible without the help of many friends and colleagues. First and foremost, I thank my committee: Amy McGovern (University of Oklahoma [OU]) for keeping me on track, providing continual input to my research and academic career, and giving me the opportunity to teach students about machine learning; Travis Smith (OU and Cooperative Institute for Mesoscale Meteorological Studies [CIMMS]) for providing his meteorological expertise and giving me the opportunity to attend many conferences, where I shared my work with the rest of the meteorology community; and Michael Richman (OU) for keeping me grounded in statistical reality.

I thank Valliappa Lakshmanan (Google) for helping me to secure this project and providing crucial input during the early stages; David John Gagne II (National Center for Atmospheric Research) for his input during many lab meetings and late-night Facebook conversations; Jonathan Edwards-Opperman (OU) for his input during many late-night roommate conversations; Darrel Kingfield and Kiel Ortega (CIMMS) for their technical support, without which I would not have had the computing resources to finish this project; Elizabeth Smith and Veronica Fall (OU) for donating their time to listen my practice defence presentations; and Mallory Row (National Centers for Environmental Prediction) for guiding me through the tangled web of bureaucracy involved in the thesis-writing and defence process. I also need to thank the last three for the tremendous amount of moral support that they offered.

For their moral and spiritual guidance in this endeavour, I would like to thank Canadian documentarians Bob McKenzie, Doug McKenzie, and Red Green; Ricky, Julian, Bubbles, Steve French, trailer-park supervisor Jim Lahey, and assistant trailer-park supervisor Randy of Sunnyvale Trailer Park; and hockey player and entrepreneur Tim Horton.

Table of Contents

List of Tables

List of Figures

Abstract

Thunderstorms, including straight-line (non-tornadic) winds, cause an average of over 100 deaths and $10 billion of insured damage per year in the United States. In the past decade machine learning has led to significant improvements in the prediction of other convective hazards, such as tornadoes, hail, lightning, and convectively induced aircraft turbulence. However, very few studies have used machine learning specifically to predict damaging straight-line winds. We have developed machine-learning models to predict the probability of damaging straight-line wind, defined as a gust $\geq$ 50 kt (25.72 m s^{-1}), for a given storm cell. Predictions are made for three buffer distances around the storm cell (0, 5, and 10 km) and five lead-time windows ($[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes).

Three types of data are used to train models: radar images from the Multi-year Reanalysis of Remotely Sensed Storms (MYRORSS); atmospheric soundings from the Rapid Update Cycle (RUC) model and North American Regional Reanalysis (NARR); and near-surface wind observations from the Meteorological Assimilation Data Ingest System (MADIS), Oklahoma Mesonet, one-minute meteorological aerodrome reports (METARs), and National Weather Service local storm reports. Radar images are used to determine the structural and hydrometeorological properties of storm cells, while soundings are used to determine properties of the near-storm environment, which are important for

storm evolution. Both of these data types are used to create predictor variables. Meanwhile, near-surface wind observations are used as verification data (to determine which storm cells produced damaging straight-line winds).

For each buffer distance and lead-time window, we experiment with five machine-learning algorithms: logistic regression, logistic regression with an elastic net, feed-forward neural nets, random forests, and gradient-boosted tree (GBT) ensembles. Forecast probabilities from each model are calibrated with isotonic regression, which makes them more reliable. Forecasts are verified mainly with three numbers: area under the receiver-operating-characteristic curve (AUC), maximum critical success index (CSI), and Brier skill score (BSS). AUC and maximum CSI range from $[0, 1]$, where 0 is the worst score and 1 is a perfect score. BSS ranges from $(-\infty, 1]$, where $-\infty$ is the worst score; 1 is a perfect score; and > 0 means that the model is better than climatology. Models are ranked by AUC. The best model (for a buffer distance of 0 km and lead time of $[15, 30]$ minutes) has an AUC of 0.996, maximum CSI of 0.99, and BSS of 0.88. The worst model (for a buffer distance of 10 km and lead time of $[60, 90]$ minutes) has an AUC of 0.89, maximum CSI of 0.20, and BSS of 0.12. All models outperform climatology.

Finally, for each buffer distance and lead-time window, we use three methods to select the most important predictor variables: sequential forward selection, J-measures, and decision trees.

Chapter 1

Introduction

In an average year in the United States, thunderstorms cause over 100 deaths and $10 billion of insured damage[1]. Some of these losses are caused by non-tornadic (straight-line) wind events, such as downbursts, gust fronts, bow echoes, and derechoes. Although information is not available on the exact losses caused by straight-line winds versus tornadoes, damaging straight-line winds are much more common than tornadoes[2], so there is reason to believe that they cause a comparable amount of losses.

Both thunderstorm occurrence and near-surface wind are non-linearly related to many meteorological factors, which cannot be predicted adequately by numerical weather prediction (NWP) models. This is because NWP models (a) do not solve the equations of motion exactly and (b) cannot resolve motions at a scale below their grid spacing. The finest-scale operational models [*e.g.*, the 4-km North American Mesoscale (NAM-4km) (Aligo et al., 2014), High-Resolution Rapid Refresh (HRRR) (Clark et al., 2012), and Advanced Research Weather Research and Forecasting (WRF-ARW) (Skamarock and Klemp, 2008) models] have a grid spacing of 3-4 km, which allows them to predict thunderstorm occurrence with some fidelity, but they have trouble predicting thunderstorm-related

[1] `http://www.iii.org/fact-statistic/catastrophes-us`

[2] `http://www.nssl.noaa.gov/education/svrwx101/wind`

phenomena such as damaging straight-line wind. This is why meteorologists often use machine learning to predict such phenomena.

Machine learning is a type of artificial intelligence in which computers learn knowledge that has not been explicitly programmed. Usually the goal is to predict one dependent variable y based on many predictor variables x_j. Machine learning consists of three steps.

1. Training. The goal is to learn a relationship between the x_j and y that optimizes some performance measure (*e.g.*, mean absolute error, mean squared error, classification accuracy). The performance measure is computed by comparing the predicted y-values to actual y-values. This relationship is learned from many examples, which comprise the training set, and is called a "model".
2. Testing. The model is tested on a new set of examples, which should be (a) independent of the training set and (b) similar to examples that the model will encounter in the future. If these criteria are met, performance on the testing set should be similar to future performance (*e.g.*, when the model is deployed in a forecasting office).
3. Deployment. Here the model is used to predict new examples.

Some advantages of machine learning are as follows.

1. In the deployment phase, a machine-learning model (unlike human reasoning) is determinstic. In other words, given the same predictor values x_j, it will always predict the same value y.
2. Deployment time is usually very fast ($\ll 1$ second to predict a new example, whereas running an NWP model takes ~1 hour).

3. It can predict the outcome of a physical process without fully understanding said process (and without the developers or users fully understanding said process)[3].
4. Output from NWP models can be used to create predictors, which helps to keep machine-learning predictions realistic.
5. Certain machine-learning and variable-selection methods can be used to gain insight into the underlying physical processes.

The goal of this project is to predict the occurrence of damaging straight-line wind for a given storm cell. "Damaging wind" is defined as a gust $\geq$ 50 kt (25.72 m s^{-1}), which is the National Weather Service's (NWS) threshold for severe thunderstorms[4]. Specifically, we develop models to predict damaging straight-line wind at three buffer distances around the storm cell (0, 5, and 10 km) and five lead-time windows ($[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes).

Our hypothesis was that for each combination of buffer distance, lead-time window, and forecast probability, we could produce reliable forecasts that outperform climatology. We could not directly compare our performance to previous models, because previous models either (a) forecast many thunderstorm hazards at once, rather than focusing specifically on straight-line wind, or (b) forecast straight-line wind only for storms that are known to be severe.

We make the following improvements over previous studies: (a) using station-based wind observations, rather than only human reports; (b) using multiple

[3]This can also be a disadvantage, because it can lead to over-reliance on machine learning and a lack of emphasis on physical understanding, which is why we use variable selection to aid physical understanding (see item 5).

[4]`http://www.spc.noaa.gov/misc/about.html`

data sources to create predictors; (c) predicting only straight-line wind, separate from other thunderstorm hazards; (d) using composited, rather than single-radar, data; (e) using advanced machine-learning algorithms with probability calibration; and (f) using a large number of predictors.

Chapter 2

Related Work

Section 2.1 explains the physical mechanisms behind damaging straight-line convective winds; the remaining subsections are a review of machine-learning applications in radar and convective meteorology. First, we review expert systems, which are the earliest and best-known machine-learning applications in said field. Then we review other machine-learning methods used for radar and convective meteorology. Finally, we review applications of machine learning to predict damaging straight-line convective winds specifically.

2.1 Physical Mechanisms Behind Damaging Straight-line Convective Winds

There are three general types of damaging straight-line convective wind[1], which will be described in turn: downbursts (including macrobursts and microbursts), gust fronts, and bow echoes (including derechoes).

A downburst (Fujita, 1990; Ferrero et al., 2014; Jesson et al., 2013) is an area of strong wind caused by a downdraft hitting the surface and spreading out horizontally (Figure 2.1). Two mechanisms are primarily responsible for downbursts. The first is evaporative cooling, which occurs when hydrometeors (particles of rain, snow, hail, etc.) fall into subsaturated air (air with less than the saturation water-vapour content) below the cloud base. The hydrometeors evaporate (if liquid) or sublimate (if ice), which is an endothermic reaction, requiring heat from the surrounding air. This cools the surrounding air, making it denser and causing it to sink. The second mechanism is precipitation drag, which occurs when falling hydrometeors exert a frictional force on the surrounding air, thus accelerating the air downward. These two mechanisms can occur at the same time and have additive effects.

Downbursts are generally characterized in two ways: macroburst vs. microburst and dry vs. wet. A macroburst has a horizontal area > 4 km^2, and a microburst has an area < 4 km^2. A wet downburst is accompanied by heavy

[1] `http://www.nssl.noaa.gov/education/svrwx101/wind/types` splits damaging straight-line convective wind into five types: downbursts, microbursts, gust fronts, derechoes, and haboobs. However, a microburst is a special type of downburst, and a derecho is a bow echo or series thereof. This section is organized by the general category, rather than subcategory. Also, haboobs (walls of dust carried along with a gust front) are highly dependent on local land cover and moisture (*i.e.*, there must be dry, fine soil at the surface), so they will not be discussed in this document.

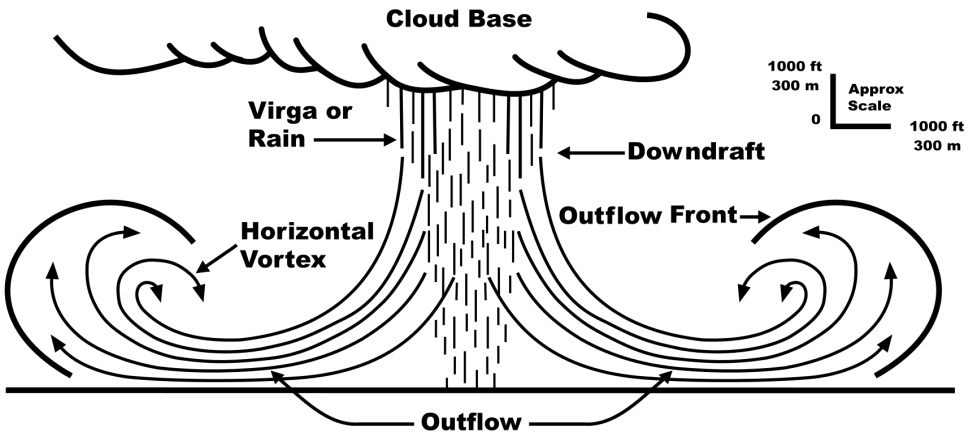

Figure 2.1: Schematic of a downburst. The "outflow front" is the same as the gust front, or outflow boundary, discussed in Section 2.1. Image from Federal Aviation Administration (ed. D.C. Beaudette), vectorized by FOX 52, public domain, `https://commons.wikimedia.org/w/index.php?curid=46182311`.

precipitation, which occurs when the cloud base is low enough that not all hydrometeors evaporate/sublimate before they reach the ground. A dry downburst occurs with no precipitation, because the cloud base is high enough that all hydrometeors evaporate/sublimate in the dry air below the cloud base. (Fujita, 1990) There are intermediate cases between wet and dry downbursts. The main difference between macrobursts and microbursts, other than their horizontal area, is their strength and duration[2]. A macroburst usually lasts for about 30 minutes and may produce winds up to 130 mi h^{-1} (~110 kt or 60 m s^{-1}), whereas a microburst usually lasts for about five minutes and may produce winds up to 160 mi h^{-1} (~140 kt or 70 m s^{-1}). There are no clear differences in strength or duration between wet and dry downbursts. (Fujita, 1990)

The following environmental conditions are most conducive to downbursts (Rose, 1996).

[2] `http://www.nssl.noaa.gov/education/svrwx101/wind/types`

1. Absolute static instability. This occurs when the environmental lapse rate (downward partial derivative of temperature, or $-\frac{\partial T}{\partial z}$) is greater than the dry adiabatic lapse rate (~9.8 °C km^{-1}), which allows the descending air parcel to remain cooler than the environment all the way to the ground. As long as the descending parcel is cooler than the environment, it will receive a negative buoyant force, which will increase its downward velocity.
2. Subsaturated air below the cloud base. Hydrometeors can evaporate/sublimate only in subsaturated air, so if the air is saturated, one of the two primary downburst mechanisms is lost.
3. Moist air near the surface. At a given temperature and pressure, moist air is less dense than dry air. Thus, if the near-surface air is moister, it will impart an even greater downward acceleration to the descending air parcel when the two meet.
4. Frozen precipitation above the subsaturated air. If hydrometeors fall into the subsaturated air as ice particles (*e.g.*, snow, hail, or graupel), they will remove more heat from the air, because sublimation requires more heat than evaporation.

A gust front (or "outflow boundary") (Delanoy and Troxel, 1993) occurs when strong horizontal winds in a downburst propagate away from the storm, creating a boundary between evaporatively cooled air in the downburst and warmer air in the surrounding environment (Figure 2.1). Because the two air masses have a large temperature difference (and the cooler air mass is usually drier), they also have a large density difference. The evaporatively cooled air, which is denser, displaces the surrounding air and forces it upward. This often creates a shelf cloud along the leading edge of the storm. More importantly, though, convergence along the gust front causes turbulence and wind shear, which may be extremely hazardous to aircraft.

A bow echo (Johns, 1993) is a bow-shaped line of storm cells, usually associated with damaging straight-line wind along its leading edge. There are three steps in the formation of a bow echo.

1. A downburst, produced by a single storm cell, propagates away from the cell as a gust front (Figure 2.2). The strongest winds along the gust front are in the middle (near the core of the downburst), which forces the gust front into a bow shape.
2. Convergence along the gust front leads to new convective development, which leads to a bow-shaped line of storm cells (Figure 2.3).
3. Divergence at the surface, associated with the downburst and gust front, is balanced by convergence in the mid-troposphere. Convergence is concentrated near the apex of the bow (dashed orange line in Figure 2.4), where it is fed by a rear-inflow jet (Figure 2.5). The rear-inflow jet descends at the leading edge of the system, which produces more downbursts. This strengthens the gust front, which strengthens convergence and rising motion ahead of the gust front. Rising motion ahead of the gust front balances downward motion behind the gust front (in the rear-inflow jet), which closes the positive-feedback loop.

Eventually, either (a) the bow echo moves into an unfavourable environment (insufficient moisture, instability, wind shear, etc.) or (b) the apex pushes out so far that the system becomes distorted, causing the circulation to break down.

Finally, a derecho is a long-lived bow echo or series thereof, usually associated with a mesoscale convective system (MCS) (Coniglio et al., 2004). An MCS is a cluster of storm cells, organized on a scale larger than that of the individual storm cells but smaller than that of an extratropical cyclone (synoptic-scale low-pressure system) (Zipser, 1982). Winds in a derecho may reach 130 mi h^{-1} (~110

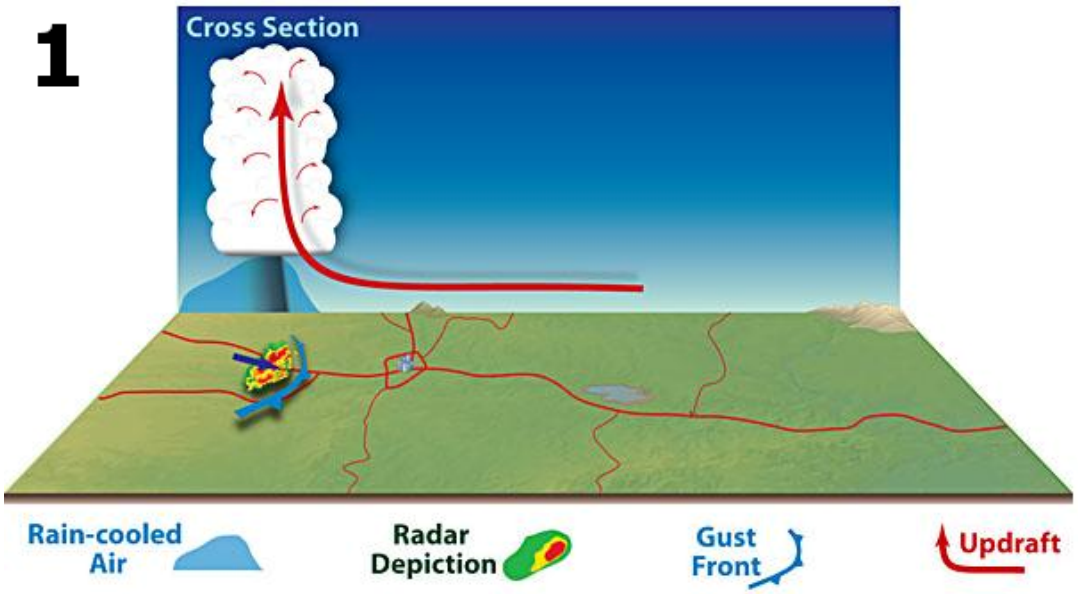

Figure 2.2: Step 1 in the formation of a bow echo. A downburst, produced by a single storm cell, propagates away as a gust front. Image from `http://www.spc.noaa.gov/misc/AbtDerechos/derechofacts.htm`, modified from illustration by Dennis Cain.

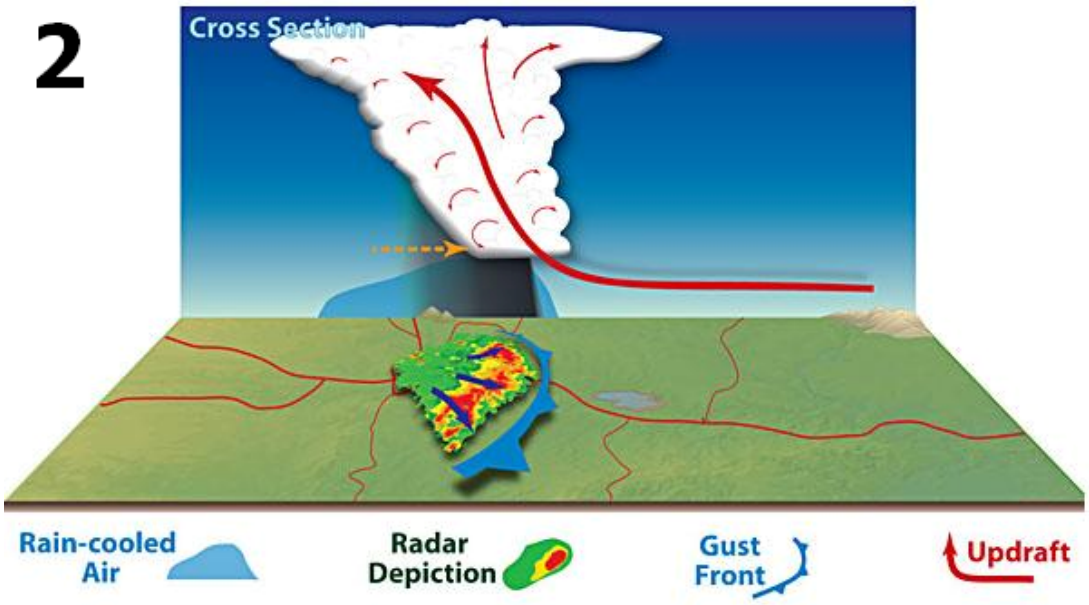

Figure 2.3: Step 2 in the formation of a bow echo. Convergence along the bow-shaped gust front leads to new storm initiation. Image from `http://www.spc.noaa.gov/misc/AbtDerechos/derechofacts.htm`, modified from illustration by Dennis Cain.

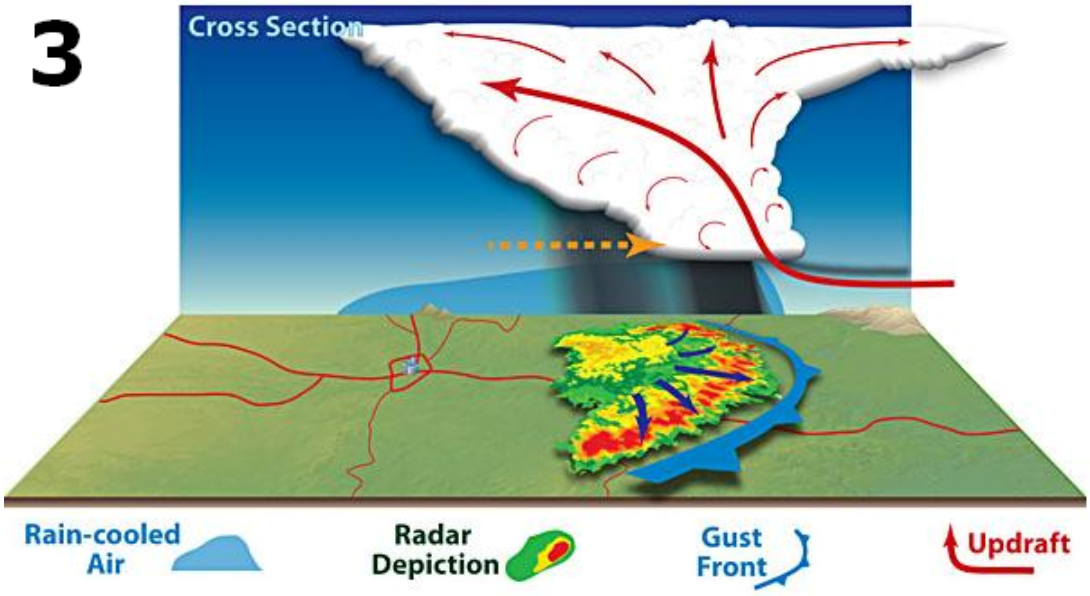

Figure 2.4: Step 3 in the formation of a bow echo. Divergence at the surface is balanced by convergence in the mid-troposphere (near the head of the dashed orange line). Image from `http://www.spc.noaa.gov/misc/AbtDerechos/derechofacts.htm`, modified from illustration by Dennis Cain.

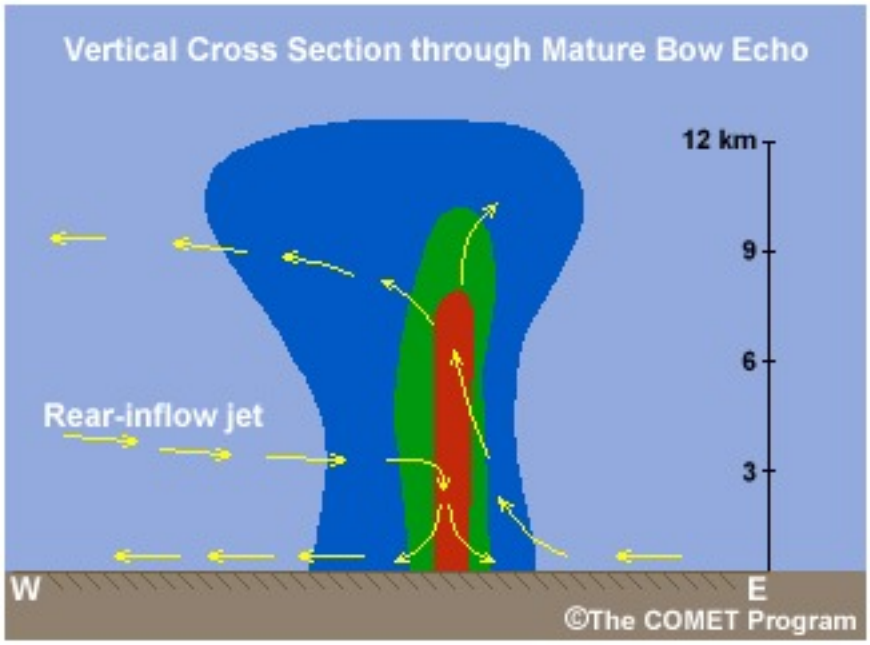

Figure 2.5: Vertical cross-section of a mature bow echo. Rising motion ahead of the gust front (associated with convection) is balanced by downward motion behind the gust front (in the rear-inflow jet). Image from `http://www.meted.ucar.edu/mesoprim/severe2/print_version/_p_4.3BE-Features.htm`.

kt or 60 m s^{-1})[3]. Traditionally, a derecho has also been defined by the following criteria (Corfidi et al., 2016), where "severe wind" is defined as a gust $\geq$ 50 kt or one that produces significant damage. "F1 wind" is defined as a gust $\geq$ 65 kt or one that produces damage similar to an F1 tornado.

1. The severe-wind area must have a major axis $\geq$ 400 km.
2. Individual severe-wind reports must exhibit a non-random pattern. Specifically, reports must occur in a swath or series thereof (which may correspond to a bow echo or series thereof).
3. The severe-wind area must contain $\geq$ 3 reports of F1 wind, separated by a distance of $\geq$ 64 km.
4. There may be no more than 3 hours between successive severe-wind reports.

Corfidi et al. (2016) recently proposed the following definition, which is more restrictive and physically based: "A family of damaging downburst clusters associated with a forward-propagating, mesoscale convective system (MCS) that, during part of its existence, displays evidence of one or more sustained bow echoes with mesoscale vortices and/or rear-inflow jets." This new definition also includes the following criteria.

1. The severe-wind area must have a major axis $\geq$ 650 km and minor axis $\geq$ 100 km along most of the major axis.
2. Severe-wind reports are counted only after preliminary storms (*e.g.*, the storm cell shown in Figure 2.2, which initiates a bow echo) have organized themselves into a cold-pool-driven MCS.

Favourable conditions for derechoes are more difficult to discuss, because they occur in a wide variety of environments. Since derechoes are much longer-lived and larger-scale than individual downbursts or gust fronts, they may be

[3] `http://www.spc.noaa.gov/misc/AbtDerechos/derechofacts.htm`

strongly influenced by synoptic-scale forcing. Thus, when analyzing derecho environments, it is common to separate derechoes into categories based on the amount of synoptic-scale forcing. Coniglio et al. (2004) separated derechoes into those with weak, moderate, and strong synoptic-scale forcing, based on the Q-vector divergence, which is a proxy for synoptically forced vertical motion (Section 5.7.4 of Bluestein, 1992). Beyond the conditions associated with individual downbursts (discussed earlier), Coniglio et al. (2004) found that the following conditions are associated with each type of derecho.

1. Weakly and moderately forced derechoes tend to have very high CAPE (mean of ~3000 J kg^{-1} for cases analyzed in Coniglio et al., 2004). Strongly forced derechoes have more moderate CAPE values (mean of ~1400 J kg^{-1} with many cases $<$ 1000 J kg^{-1}).
2. Strongly forced derechoes tend to have very high low-level shear (mean of ~30 kt from 0-1 km above ground level [AGL]) and moderate low–mid-level shear (mean of ~46 kt from 0-5 km AGL). Weakly and moderately forced derechoes tend to have much weaker low-level shear (means of ~13 and ~16 kt from 0-1 km AGL, respectively) but moderate low–mid-level shear (both with means of ~37 kt from 0-5 km AGL, which is more similar to the strongly forced category).
3. Strongly forced derechoes tend to have high near-surface relative humidity (RH) and low RH from the near-surface to mid-troposphere. Specifically, the median profile has an RH $>$ 80% from 0-1 km AGL and ~40% around 3 km. Meanwhile, weakly and moderately forced derechoes tend to have moderate near-surface RH (median values $>$ 60% and 70%, respectively, from 0-1 km AGL) and low RH from the near-surface to mid-troposphere (minimum of ~40% around 3 km, as for the strongly forced category).

In summary, strongly forced derechoes tend to have greater low-level shear, smaller CAPE, and smaller near-surface relative humidity. Weakly and moderately forced derechoes tend to have smaller low-level shear, greater CAPE, and greater near-surface relative humidity. Not surprisingly, weakly and moderately forced derechoes are more common during the warm season (May-August), whereas strongly forced derechoes are more common during the cool season (September-April) (Coniglio et al., 2004).

2.2 Expert Systems in Radar and Convective Meteorology

Machine learning has been used to post-process radar data and predict thunderstorm-related phenomena since the early 1980s. The first machine-learning techniques used for this purpose were expert systems, which attempt to explicitly represent human knowledge in an objective form. Thus, unlike most machine learning, the goal of expert systems is knowledge representation and deployment, rather than knowledge discovery. Expert systems are usually encoded as a set of if-then rules, for which the thresholds may be determined either *a priori* or by another machine-learning algorithm.

The expert system developed by Zrnić et al. (1985), called the Build 9.0 Mesocyclone Algorithm (B9MA), was the first mesocyclone-detection algorithm (MDA) for Next-Generation Radar (NEXRAD) (Crum and Alberty, 1993). NEXRAD is a network of ~160 radars in the continental United States (CONUS) (Figure 2.6). The B9MA is described below, to give the reader a flavour of expert systems.

1. Find areas where $\frac{\partial v_r}{\partial \theta}$ (the azimuthal gradient of radial velocity, or horizontal velocity parallel to the radar beam) is either consistently increasing or decreasing. These are called "one-dimensional (1-D) shear areas".
2. For each 1-D shear area, calculate five predictors: start/end azimuth, start/end radial velocity, and range (distance from radar).
3. Eliminate 1-D shear areas below the low angular-momentum threshold (50 km m s^{-1}).
4. Eliminate 1-D shear areas below the low shear threshold (2 m s^{-1} km^{-1}).
5. Eliminate 1-D shear areas below the high angular-momentum threshold (150 km m s^{-1}) and high shear threshold (4 m s^{-1} km^{-1}).
6. Combine 1-D shear areas into 2-D shear areas.
7. Eliminate 2-D shear areas containing less than six 1-D shear areas.
8. Eliminate asymmetric shear areas (this involves many criteria, discussed in Section 4c of Zrnić et al., 1985).

The main shortcoming of the B9MA (and many other expert systems) is that it applies many detection thresholds in series and any object (candidate mesocyclone) failing one of these thresholds is discarded. This serial approach was necessary at the time (objects were eliminated in each step, so fewer objects remained to be processed at each successive step), but it led to many false negatives and thus a low probability of detection (POD) (Equation 4.22a).

Wieler (1986) developed an expert system to detect "shear areas," which include both mesocyclones and other phenomena. This system, called the Mesocyclone Tornadic Vortex Signature Detection Algorithm (MTDA), employed a similar procedure to the B9MA. However, the MTDA had the following advantages.

1. Joined shear areas both vertically and horizontally (into 3-D features), whereas the B9MA joined areas only horizontally (into 2-D features).

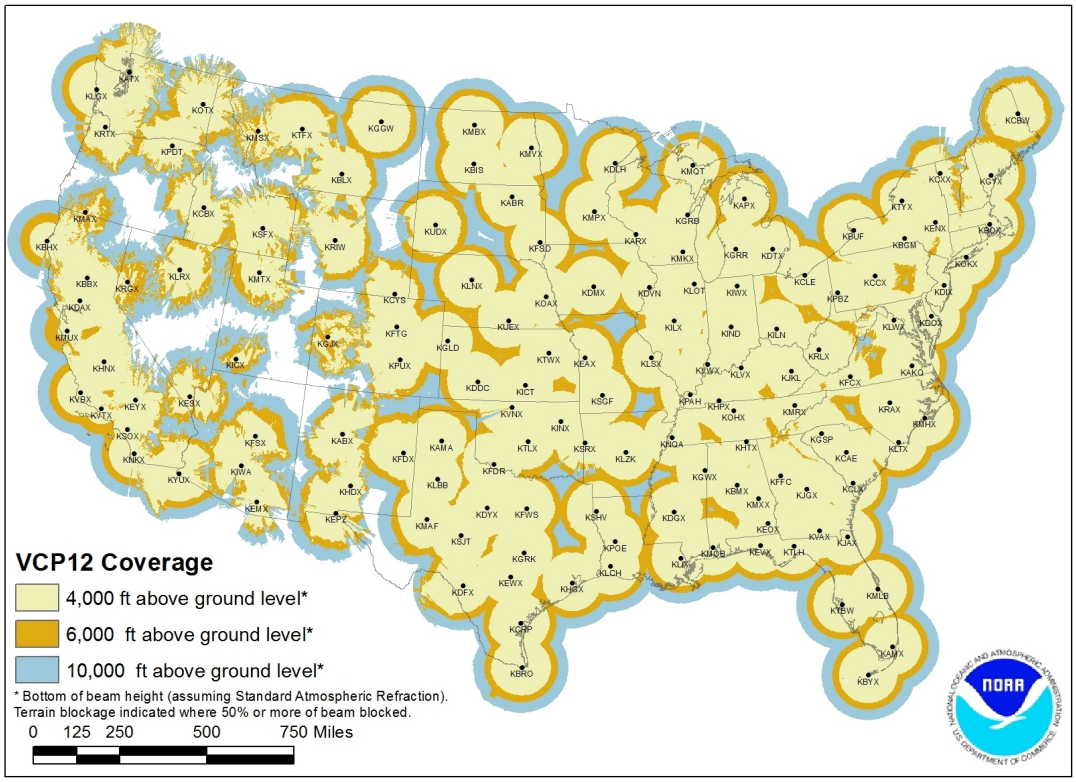

Figure 2.6: NEXRAD sites in the CONUS. Figure obtained from `https://www.roc.noaa.gov/wsr88d/maps.aspx`.

2. Matched shear areas to storm cells (detected by another algorithm).
3. Detected vortices from the scale of a tornado-vortex signature (TVS) (~100 m) to large mesocyclone (20 km), whereas the B9MA detected only vortices with a diameter of at least ~2 km.
4. Detected both cyclonic and anticyclonic shear areas, whereas the B9MA detected only cyclonic shear.
5. Classified shear areas into six types: asymmetric shear, convergence, divergence, couplet, mesocyclone, and TVS.

Also, Campbell and Olson (1987) created an expert system to detect low-altitude wind-shear hazards, primarily microbursts and gust fronts. They made their system more robust by (a) using several feature-detection methods, each with different strengths and weaknesses, and (b) including a confidence factor for each detected feature, based on the strength of the feature and the quality of radar data used to identify said feature. Thus, by the late 1980s, expert systems had become very advanced and were used to detect many thunderstorm-related hazards.

One of the best-known expert systems in radar meteorology is the National Severe Storms Laboratory's mesocyclone-detection algorithm (NSSL MDA) (Stumpf et al., 1998). This system was also procedurally similar to the B9MA but made the following improvements.

1. Did not immediately discard 1-D shear segments below a certain strength threshold.
2. Used much lower (less restrictive) strength thresholds.
3. Checked for vertical and temporal continuity of candidate mesocyclones.

Due to improvements 1 and 2, the NSSL MDA had a much higher POD. However, it also had a much higher probability of false detection (POFD) (Equation 4.22c). This POFD was greatly reduced in future studies by applying neural nets

to the NSSL MDA objects (detected mesocyclones) to determine whether or not they were "significant" (severe-weather-producing) mesocyclones (*e.g.*, Marzban and Stumpf, 1996, 1998; Marzban, 2000). In convective and radar meteorology, expert systems have also been combined with fuzzy logic (Lakshmanan and Witt, 1997; Pal et al., 2006), genetic algorithms (Lakshmanan, 2000), other applications of neural nets (Lakshmanan et al., 2010; Lakshmanan et al., 2014), clustering (fuzzy c-means in Pal et al., 2006; K-means in Lakshmanan et al., 2010), and random forests (Clark et al., 2015).

2.3 Other Machine Learning in Radar and Convective Meteorology

Analogue methods have been used in meteorology since at least the early 1910s (Bowie and Weightman, 1914) and have probably been used informally for much longer. The main idea behind analogue methods is that, to predict the outcome (dependent variable) of a new event, one can use the outcomes of previous events with similar characteristics (predictor variables). For example, to predict the maximum wind speed produced by a new storm cell, one could look at the maximum wind speeds of previous storm cells with similar motion vectors, shapes, radar signatures, etc.

Analogue methods in machine learning can be split into two categories: clustering and nearest-neighbour methods. For a clustering method, previous events (those in the training set) are grouped into several clusters (often called "prototypes" or "map types" in meteorology). Each event is defined by a predictor vector $\vec{x}$. To predict the outcome of a new event $\vec{x}_{new}$, the nearest cluster C is found (*e.g.*, that whose centroid $\vec{x}_C$ minimizes the Euclidean distance with $\vec{x}_{new}$). Then the outcomes of previous events in cluster C are used to predict

the outcome of the new event (*e.g.*, by taking the mean or median of all outcomes in cluster C). For a nearest-neighbour method, previous events (those in the training set) are not pre-assigned to clusters. Instead, for a new event $\vec{x}_{new}$, the k nearest neighbours (*e.g.*, the k previous events whose predictor vectors $\vec{x}$ minimize the Euclidean distance with $\vec{x}_{new}$) are found and their outcomes are used to predict the outcome of the new event.

In radar and convective meteorology, Baldwin et al. (2005) used hierarchical clustering to classify precipitation areas as convective-linear, convective-cellular, or stratiform. Gagne et al. (2009) used K-means clustering to segment a radar image into contiguous precipitation areas. Also, Lakshmanan et al. (2010) and Lakshmanan et al. (2014) used K-means clustering to segment a map of radar-echo classifications (either precipitation or non-precipitation) into relatively uniform areas.

Linear regression has been used in weather prediction since at least the early 1950s (Malone, 1955). Linear regression predicts the dependent variable y based on a linear combination of predictor variables x_j, using the following equation.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_N x_N = \beta_0 + \sum_{j=1}^{N} \beta_j x_j \tag{2.1}$$

The coefficients β_j are learned by a statistical algorithm, usually gradient descent (Section 4.4.3 of Mitchell, 1997). Logistic regression is similar, except that the dependent variable is transformed to $\ln(\frac{y}{1-y})$ (see Figure 4.1). This makes logistic regression ideal for predicting probabilities, because y is constrained to $[0, 1]$ (see Section 4.1.1.1).

However, neither linear nor logistic regression have found many uses in radar and convective meteorology, because relationships between thunderstorm phenomena and other weather variables are highly non-linear. Some exceptions are Kitzmiller et al. (1995), which used linear regression to forecast the probability of severe weather (damaging straight-line winds, tornadoes, or hail) for a storm

cell; Billet et al. (1997), which used linear regression to forecast maximum hail size, and logistic regression to forecast large-hail probability, from a storm cell; and Mecikalski et al. (2015), which used logistic regression to forecast the probability of convective initiation (first appearance of reflectivity $\geq$ 35 dBZ at the -10 °C temperature level) within a cumulus cloud.

Although the first objective decision-tree-learning method was not developed until the mid-1980s (Quinlan, 1986), subjective (human-derived) decision trees have been used in meteorology since at least the mid-1960s (Chisholm et al., 1968). At each node (Figure 4.4), a binary question is asked for one predictor variable x_j. For each event (defined by a predictor vector $\vec{x}$), if the answer is yes, $\vec{x}$ is sent down the right branch; if the answer is no, $\vec{x}$ is sent down the left branch. This continues until $\vec{x}$ reaches a leaf node n, from which there are no further branches. Then the outcomes of previous events (those in the training set) reaching leaf node n are used to predict the outcome of the new event. Decision trees can be used for either classification or regression (prediction of a real value, rather than discrete category).

Williams et al. (2008a) used random forests (decision-tree ensembles) to predict vertically integrated liquid for a storm cell; Williams et al. (2008b) used random forests to predict four categories (none, light, moderate, and severe) of turbulence produced by a storm cell; Gagne et al. (2009) used decision trees to classify precipitation regions into six categories (isolated pulse, isolated strong, and multicellular storms; and trailing, leading, and parallel stratiform regions); Williams (2014) used random forests to predict two categories (severe or non-severe) of turbulence from a storm cell; and Clark et al. (2015) used random forests to reduce false detections from an expert dryline-identification system.

Spatiotemporal relational probability trees (SRPTs) were developed by McGovern et al. (2008). There are two main differences between a traditional

decision tree and an SRPT. First, for a traditional decision tree each event is represented by a predictor vector $\vec{x}$, whereas for an SRPT each event is represented by many interrelated fields and objects. Second, whereas a traditional decision tree can ask only whether a single predictor x_j is above or below a certain threshold, an SRPT can ask five types of questions for each event (McGovern et al., 2013). In the following list, an "item" refers to either an object or a relation between objects.

1. Basic questions (non-spatial and non-temporal). An example is the "exists" question (does an item of type t exist?).
2. Temporal questions. An example is the temporal-ordering question (do items in group A occur in a temporal relationship with items in group B?). The seven types of temporal relationships are "before," "meets," "overlaps," "equals," "starts," "finishes," and "during".
3. Field-related questions. An example is the field-divergence question, which is relevant only for vector fields (is the {minimum, mean, maximum, standard deviation} divergence of field f ever $\geq v$?).
4. Shape-related questions. An example is the primary-shape question, which can be asked for both 2-D and 3-D objects (*e.g.*, is the primary shape of field f a rectangular prism, sphere, cylinder, or cone?).
5. Operators, which combine two questions for the same event. An example is the Boolean-and operator (does the current event answer yes to questions a and b?).

The advantage of SRPTs is their use of high-level reasoning. Human forecasters are successful because they focus on high-level objects (*e.g.*, cyclones, cold fronts, warm fronts, drylines, low-level jets, storm cells, mesocyclones, etc.), their spatiotemporal properties, and the relationships between them. SRPTs [and ensembles thereof, called spatiotemporal relational random forests (SRRFs)

(Supinie et al., 2009)] allow this spatiotemporal relational reasoning to be emulated, but unlike expert systems they allow for the discovery of new knowledge, rather than mere knowledge representation and deployment.

Gagne et al. (2012) used SRRFs to predict the probability of tornadogenesis for each storm cell, based on surface observations and NWP output; Gagne et al. (2013) used SRRFs to predict the probability of severe hail at each point in a grid, at 18–30-hour lead times, based on storm cells produced by an ensemble of convection-allowing NWP models; McGovern et al. (2014a) used SRRFs for the same problem as in Gagne et al. (2012) and to predict the probability of severe aircraft turbulence at each point in a grid; and McGovern et al. (2014b) used SRRFs to classify low-altitude rotation (a proxy for tornado occurrence) for modeled storm cells into three categories (weak, moderate, and strong).

In Gagne et al. (2013) the SRRF outperformed traditional random forests, and in McGovern et al. (2014a) it outperformed a widely used turbulence-forecasting method called Graphical Turbulence Guidance (GTG), which is based on logistic regression.

The use of fuzzy logic in meteorology dates back to at least the early 1990s (Reiter, 1991). The advantage of fuzzy logic is that it accounts for uncertainty in the forecasting process, which is valuable for a stochastic system. Although the atmosphere may be regarded as deterministic (above the scale of quantum physics), there are several reasons to consider it stochastic for the sake of weather prediction. As mentioned in Chapter 1, NWP models (a) do not solve the equations of motion exactly and (b) cannot resolve motions at a scale below their grid spacing; also, (c) our observations of the atmosphere are uncertain due to instrument error. These factors make fuzzy logic a valid approach for weather prediction.

Fuzzy logic is most often used for classification, not regression. In this context, for each predictor variable x_j and each class y_k, there is a membership function $p(y_k \mid x_j)$, indicating the degree of membership in the k^{th} class for each value of x_j. The form of this membership function (*e.g.*, linear, parabolic, trapezoidal) is usually determined *a priori*, but the function has parameters that may be learned. For example, a linear membership function (Figure 2.7) has the parameters $x_{jk}^{(1)}$ and $x_{jk}^{(2)}$, which act in the following equation.

$$p(y_k \mid x_j) = \begin{cases} \frac{x_j - x_{jk}^{(1)}}{x_{jk}^{(2)} - x_{jk}^{(1)}}, & \text{if } \frac{x_j - x_{jk}^{(1)}}{x_{jk}^{(2)} - x_{jk}^{(1)}} \in [0, 1] \\ 0, & \text{if } \frac{x_j - x_{jk}^{(1)}}{x_{jk}^{(2)} - x_{jk}^{(1)}} < 0 \\ 1, & \text{if } \frac{x_j - x_{jk}^{(1)}}{x_{jk}^{(2)} - x_{jk}^{(1)}} > 1 \end{cases} \tag{2.2}$$

To classify a new event with predictor vector $\vec{x}$, there are three steps.

1. Fuzzification. All membership functions $p(y_k \mid x_j)$ are computed.
2. Aggregation. For each class y_k, the membership functions are combined (*e.g.*, averaged) into an aggregate score.
3. Defuzzification[4]. The class with the highest aggregate score is predicted.

Lakshmanan and Witt (1997) used fuzzy logic to detect bounded weak-echo regions (BWERs) in storm cells; Lakshmanan (2000) and Pal et al. (2006) used different algorithms to tune the parameters of the membership functions in Lakshmanan and Witt (1997); Adrianto et al. (2009) used fuzzy logic to predict tornado probability over the next 30 minutes at each point in a grid; and Park et al. (2009) used fuzzy logic, along with confidence factors (one for each x_j) and discrimination efficiencies (one for each x_j and y_k), to classify radar echoes into 10 categories (ground clutter, biological, dry snow, wet snow, ice crystals, graupel, big drops, rain, heavy rain, and rain/hail).

[4]To make probabilistic predictions, this step could be replaced with a calibration procedure, in which the sum of all class probabilities is constrained to equal 1.

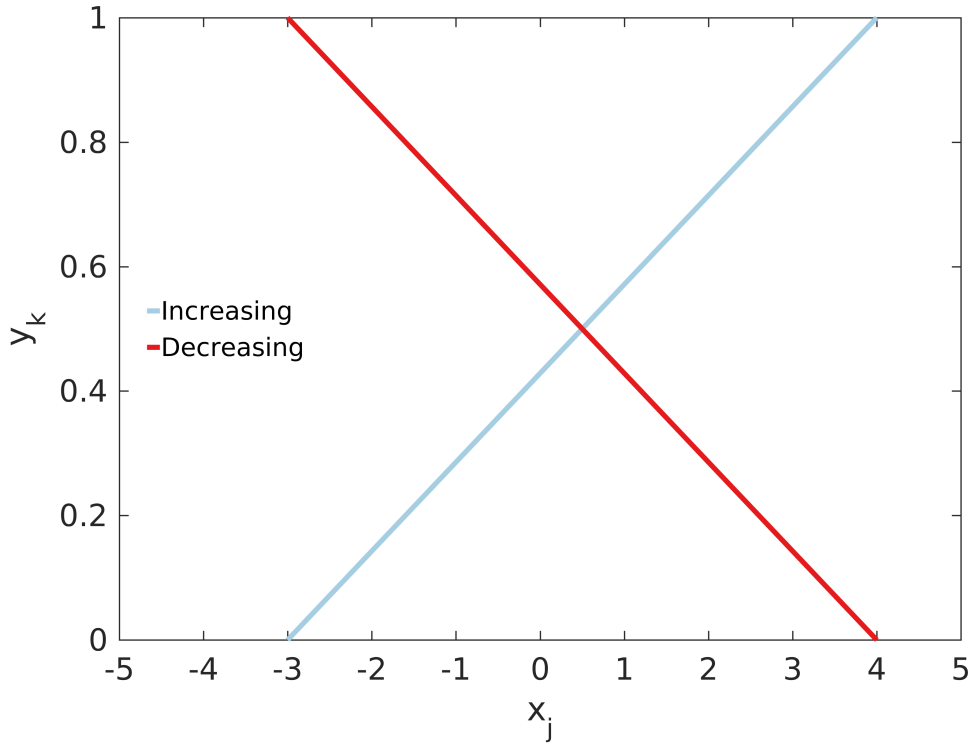

Figure 2.7: Linear fuzzy-logic membership function. x_j is a predictor variable, and y_k is the degree of membership in the k^{th} class based on the value of x_j. For the blue curve, $x_{jk}^{(1)} = -3$ and $x_{jk}^{(2)} = 4$. For the red curve, these values are opposite.

The use of feed-forward neural nets (FFNNs) in meteorology dates back to at least the late 1980s (Key et al., 1989). An FFNN contains multiple layers of neurons, with connections between neurons in subsequent layers (Figure 4.2). The first layer is the input layer, to which values of the predictor variables are fed, and the last is an output layer, which generates predictions. The input and output layers are separated by one or more internal layers, called "hidden layers". Each neuron in the k^{th} layer applies both a linear and non-linear transformation to outputs from the $(k - 1)^{\text{th}}$ layer, so the more layers there are, the more non-linear are the relationships that can be learned. Thus, the strength of an FFNN is that it can learn the complex relationships needed to predict thunderstorm-related phenomena.

Bankert (1994) used an FFNN to classify satellite imagery into 10 cloud categories (cirrus, cirrocumulus, cirrostratus, altostratus, nimbostratus, stratocumulus, stratus, cumulus, cumulonimbus, and clear); Marzban and Stumpf (1996) used FFNNs to predict tornado occurrence, and Marzban and Stumpf (1998) used FFNNs to predict damaging-wind occurrence (either straight-line or tornadic), for detected mesocyclones from the NSSL MDA; Marzban and Witt (2001) used FFNNs to predict hail size and category (small, medium, large) for objects from the NSSL hail-detection algorithm; Anagnostou (2004) used FFNNs to classify radar echoes as either stratiform or convective precipitation; Lakshmanan et al. (2010) used two FFNNs in series, and Lakshmanan et al. (2014) used four FFNNs in parallel, to classify radar echoes as either precipitation or not; and Manzato (2013) used an FFNN ensemble to predict hail occurrence and size.

2.4 Using Machine Learning to Predict Damaging Straight-line Convective Winds

To our knowledge, four previous studies have used machine learning to predict damaging straight-line convective winds. Kitzmiller et al. (1995) developed the Severe Weather Potential (SWP) algorithm for NEXRAD, which predicts the probability of severe weather (damaging straight-line winds, tornadoes, or hail) for a storm cell within the next 20 minutes. They used linear regression with five predictors (where VIL is vertically integrated liquid): VILWGT ([the number of 4-by-4-km boxes in the storm cell with VIL $\geq$ 10 kg m^{-2}] $\times$ [maximum VIL in the storm cell]), SVG10, SVG15, SVG20, and SVG25 (the number of radar pixels in the storm cell with VIL $\geq$ 10, 15, 20, and 25 kg m^{-2} respectively). The dependent variable is called the Severe Weather Potential (SWP) and varies from $[0, 40]$ in the results shown. In general, the SWP may vary over a wider range.

Marzban and Stumpf (1998) applied a feed-forward neural net to mesocyclone objects from the NSSL MDA, in order to predict the probability of all damaging winds (both straight-line and tornadic). They used 23 predictor variables, all of which are radar-derived and listed in Section 4 of Marzban and Stumpf (1996). They compared two classification schemes (*i.e.*, two ways of combining values from the FFNN's output nodes), as well as different numbers of hidden neurons and sampling distributions of the training data.

Alexiuk et al. (1999) used four algorithms (feed-forward neural nets, fuzzy c-means clustering, decision trees, and k nearest neighbours) to classify storm cells into four categories: those that produce damaging straight-line winds, tornadoes, hail, and heavy rain. They used 22 predictors, all of which are radar-derived and

listed in their Table 1. For verification data, they used reports from meteorologists and ground observers. They ran each algorithm separately and found that fuzzy c-means clustering produced the best results.

Finally, Cintineo et al. (2014) used naïve Bayes to predict the probability of severe weather (damaging straight-line winds, tornadoes, or hail) for each storm cell. They used five predictor variables: maximum estimated hail size (MESH), derived from radar data; most unstable convective available potential energy (MUCAPE) and effective-layer bulk shear (EBS), derived from the Rapid Refresh (RAP) model; and $\Delta\epsilon_{tot}$ (temporal derivative of 11-μm cloud emissivity at the tropopause) and Δ_{ice} (temporal derivative of cloud-ice fraction, or "glaciation rate"), derived from satellite data. They used NWS local storm reports, severe-thunderstorm warnings, and tornado warnings as verification data. However, they omitted storm cells that produced only straight-line-wind reports, thus de-emphasizing straight-line wind in favour of tornadoes and hail.

Shortcomings of these studies are listed below.

1. All used only human observations and/or severe-weather warnings as verification data. Human observations have a large number of false negatives (because severe weather often occurs in sparsely populated areas and most people do not report severe weather), and warnings have a large number of both false negatives and false positives. (See Section 4.3.3 for the formal definitions of "false positive" and "false negative".)
2. Kitzmiller et al. (1995), Marzban and Stumpf (1998), and Alexiuk et al. (1999) used only radar-derived predictor fields. It has been generally accepted that the quality of machine-learning predictions increases when multiple data sources are used (*e.g.*, Gagne et al., 2012, 2013; Cintineo et al., 2014; Williams, 2014; Hwang et al., 2015; Mecikalski et al., 2015).

3. Kitzmiller et al. (1995) and Cintineo et al. (2014) predicted severe weather in general, rather than specific phenomena. Marzban and Stumpf (1998) predicted damaging wind in general, rather than damaging straight-line wind. Although Alexiuk et al. (1999) specifically predicted damaging straight-line wind, they had no null cases (storms not associated with any severe weather). Thus, they predicted only the conditional probability of damaging straight-line wind, given that the storm produces severe weather. This would not be very useful in an operational environment.
4. Kitzmiller et al. (1995), Marzban and Stumpf (1998), and Alexiuk et al. (1999) used single-radar data, which has many quality issues that are alleviated by compositing data from nearby radars.
5. Kitzmiller et al. (1995) and Cintineo et al. (2014) used very simple machine-learning algorithms and did not compare with more advanced algorithms.
6. All used very few predictors (23 at the most).

This study addresses all of the above shortcomings.

Chapter 3

Data Sources and Processing

3.1 Data Sources

We use three types of data, listed in Table 3.1. Radar images, from the Multi-year Reanalysis of Remotely Sensed Storms (MYRORSS) (Ortega et al., 2012), and model soundings, from the Rapid Update Cycle (RUC) (Benjamin et al., 2004) and North American Regional Reanalysis (NARR) (Mesinger et al., 2006), are used to create predictors for the "event" (wind gust $\geq$ 50 kt). Near-surface wind observations from the Meteorological Assimilation Data Ingest System (MADIS) (McNitt et al., 2008), Oklahoma Mesonet (McPherson et al., 2007), one-minute meteorological aerodrome reports (METARs)[1], and NWS local storm reports[2] are used to determine when and where the event occurred.

3.1.1 Radar Images

In general, radar is used to determine the structural and hydrometeorological properties of storm cells. A single-polarization radar (the type used to create the MYRORSS dataset) emits a horizontally polarized beam of microwave radiation

[1] `ftp://ftp.ncdc.noaa.gov/pub/data/asos-onemin/td6406.txt`

[2] Local storm reports are downloaded in .csv files from `http://www.spc.noaa.gov/climo/online`.

Data Type	**Sources**	**Resolution (if applicable)**	**Time Period**
Radar images	MYRORSS	0.01° (~1 km), 5 minutes	2000-11 (excluding 2009)
Model soundings	RUC	13 or 20 km, 1 hour	Apr 1994 – Apr 2012
	NARR	32 km / 3 hours	1979-present
Near-surface wind observations	MADIS	variable	July 2001 – present
	Oklahoma Mesonet	variable, 5 minutes	1994-present
	One-minute METARs	variable, 1 minute	2000-present
	NWS local storm reports	variable	1955-present

Table 3.1: Summary of datasets. Spatial coverage for the Oklahoma Mesonet includes only the state of Oklahoma. Spatial coverage for all other datasets includes the entire continental United States.

Figure 3.1: Basic radar operation. The radar emits horizontally polarized waves (along the dashed line pointing outward) and receives scattered waves from atmospheric particles.

and receives scattered waves from said beam (Figure 3.1). Two fundamental variables are calculated from the scattered wave field: reflectivity and radial velocity.

Reflectivity is defined as follows.

$$z = \int_0^\infty D^6 N(D)\,\mathrm{d}D \tag{3.1}$$

D is the diameter (mm) of a scattering particle; *N(D)* is the number concentration ($\mathrm{mm^{-1}\ m^{-3}}$) of particles with diameter D; and z is the radar reflectivity factor (RRF) ($\mathrm{mm^6\ m^{-3}}$). Note that a scattering particle is not necessarily a hydrometeor (*i.e.,* particle containing atmospheric water vapour). Thus, a hydrometeor-classification algorithm is applied in MYRORSS (Lakshmanan et al., 2010; Tang et al., 2011) to eliminate echoes from non-hydrometeors (*e.g.,* birds, insects, and

topography). Also, in practice, the above equation is discretized and transformed to decibel scale.

$$Z = 10\log_{10}(z) = 10\log_{10}\left[\sum_{k=1}^{K} {D_k}^6 N(D_k)\Delta D\right] \tag{3.2}$$

K is the number of diameter bins; D_k is the mean diameter (mm) of the k^{th} bin; $N(D_k)$ is the number concentration (mm^{-1} m^{-3}) of hydrometeors in the k^{th} bin; ΔD is the difference (mm) between successive bin centers; and Z is the radar reflectivity factor in decibel scale (dBZ). Henceforth, following meteorological convention, I will refer to Z simply as "reflectivity".

Of all variables in Equation 3.1, only the RRF, which is the sixth moment of the drop-size distribution (DSD), is measured directly by the radar. The DSD is simply the number concentration as a function of diameter, or $N(D)$ (Figure 3.2). A single-polarization radar also measures the third moment of the DSD, which is Equation 3.1 with D^3 instead of D^6. By fitting these moments to a canonical DSD (*e.g.,* the Marshall-Palmer, exponential, or gamma distribution), one can estimate the full DSD. This procedure, called a "DSD retrieval," can be used to estimate properties such as maximum hail size in the storm cell. Maximum hail size can be used to estimate the updraft speed, which is a good indicator of both storm longevity (how long the storm will last) and downburst potential (Section 2.1), which makes reflectivity useful for this study.

The equation for radial velocity involves many radiation concepts, including scattering phase, amplitude, and geometry. Thus, it is outside the scope of this document. Conceptually, the radial velocity in each pixel is the mean hydrometeor velocity parallel to the radar beam. This is similar, but not identical, to background wind velocity parallel to the radar beam. Thus, radial velocity can be used to estimate (a) the raw wind speed; (b) the strength of convergence or divergence (Figure 3.3), the latter of which may be associated with downbursts; and (c) the strength of rotation (Figure 3.4). The strength of rotation

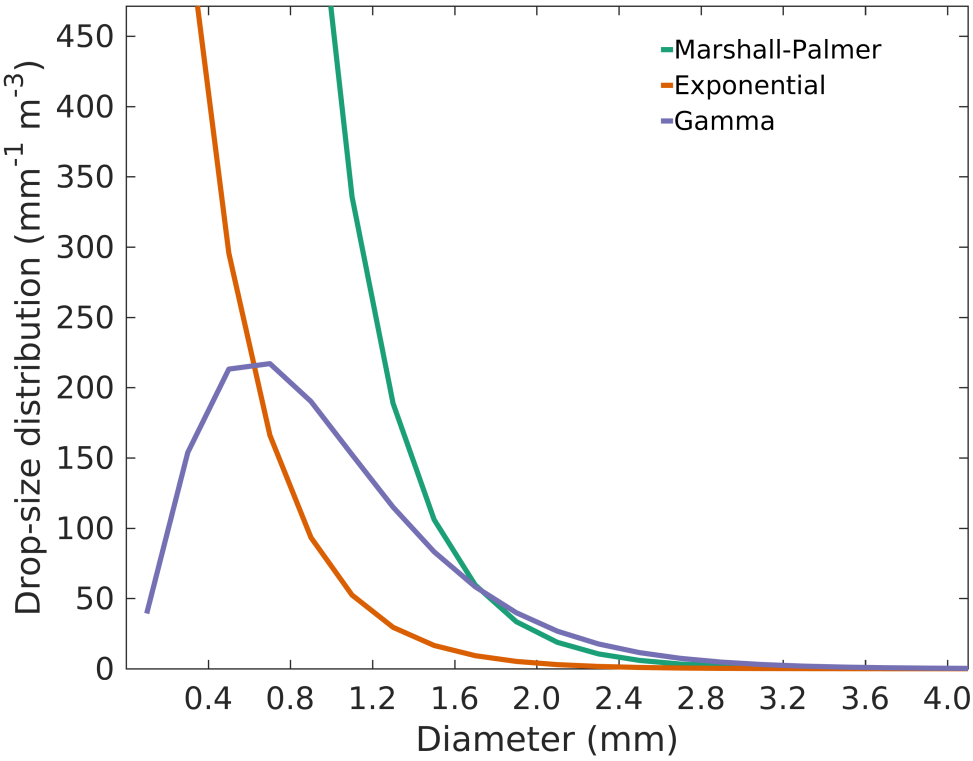

Figure 3.2: Three canonical drop-size distributions, to which radar measurements are often fit.

is a good indicator of storm longevity, because strong rotation tends to organize the updraft and downdraft into different columns (*e.g.*, in a supercell), which prevents the downdraft from extinguishing the updraft, thus allowing convection to continue.

Thus, reflectivity and radial velocity (the two fundamental radar variables) are both related to horizontal wind speed. However, the relationship with reflectivity is very indirect, and radial velocity can be used to estimate only one wind component. Another disadvantage, shared by both reflectivity and radial velocity, is that they are measured at varying heights – sometimes several kilometres – above the surface. This is because the radar beam has a "tilt angle" above the surface (Figure 3.1). Even for the lowest tilt angle (0.5° for NEXRAD), the beam reaches a height of 10 m (the standard measurement height for near-surface wind speed) at a horizontal distance of only 1.1 km from the radar. The beam reaches a height of 100 m (an order of magnitude beyond the standard measurement height) at only 10.6 km from the radar.[3] Thus, radar measurements can be used as a predictor, but not a direct indicator, of near-surface winds.

Until now, this section explains why we use radar data to predict near-surface winds and the shortcomings involved therein. We use MYRORSS specifically because it is a quality-controlled composite of all NEXRAD radars in the CONUS (Figure 2.6). MYRORSS is created by the Warning Decision Support System – Integrated Information (WDSS-II), which is a software package for the analysis and visualization of remotely sensed (mainly thunderstorm-related) weather data (Lakshmanan et al., 2007). MYRORSS composites radial velocity (Lakshmanan et al., 2006) from overlapping radars (*i.e.*, radars with overlapping horizontal ranges) to obtain two variables: low-level (0–2-km) and mid-level (3–6-km) azimuthal shear, which indicate the strength of rotation. MYRORSS composites

[3]These calculations were made with the first equation at `http://www.wdtb.noaa.gov/tools/misc/beamwidth`.

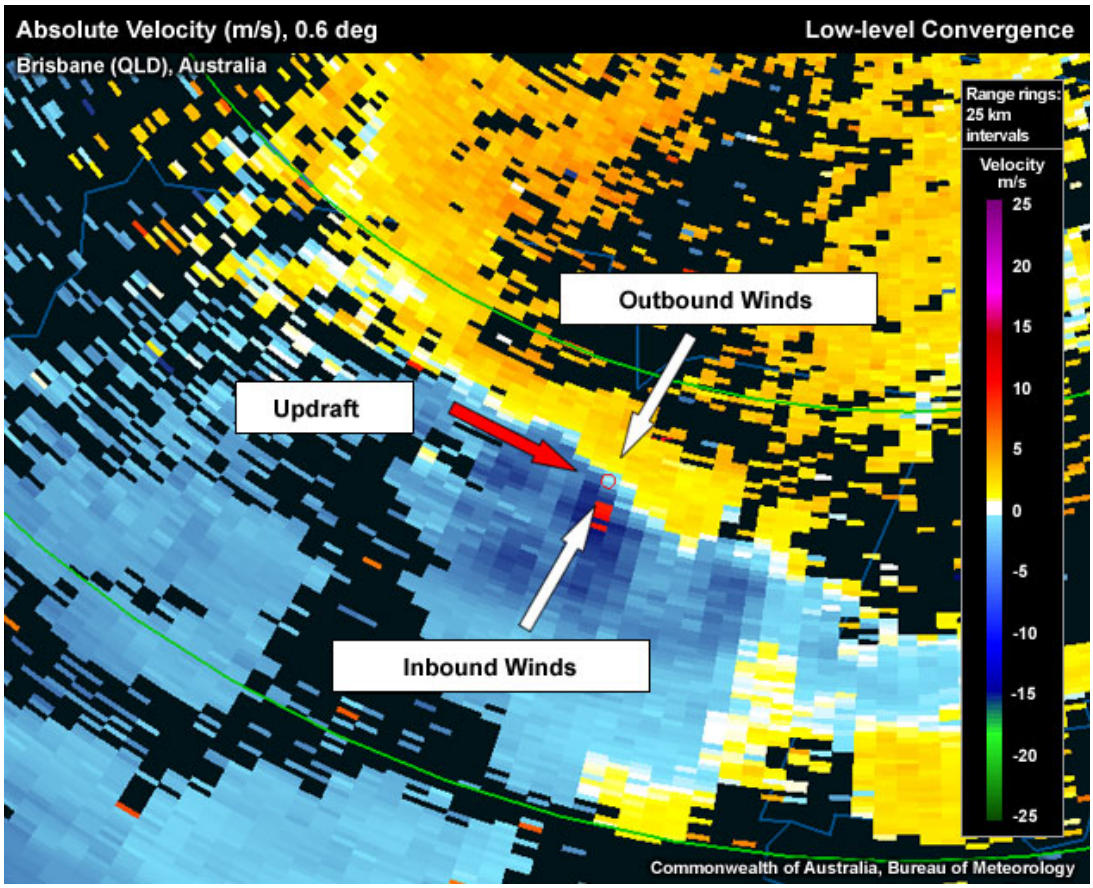

Figure 3.3: Example of convergence in radar-derived wind field. Convergence is indicated by a couplet of strong inbound and outbound velocities along the same azimuth (line pointing radially outward from the radar). For a divergent wind field, the locations of inbound and outbound velocities would be flipped. Image obtained from `http://www.meted.ucar.edu/radar/severe_signatures/index.htm`.

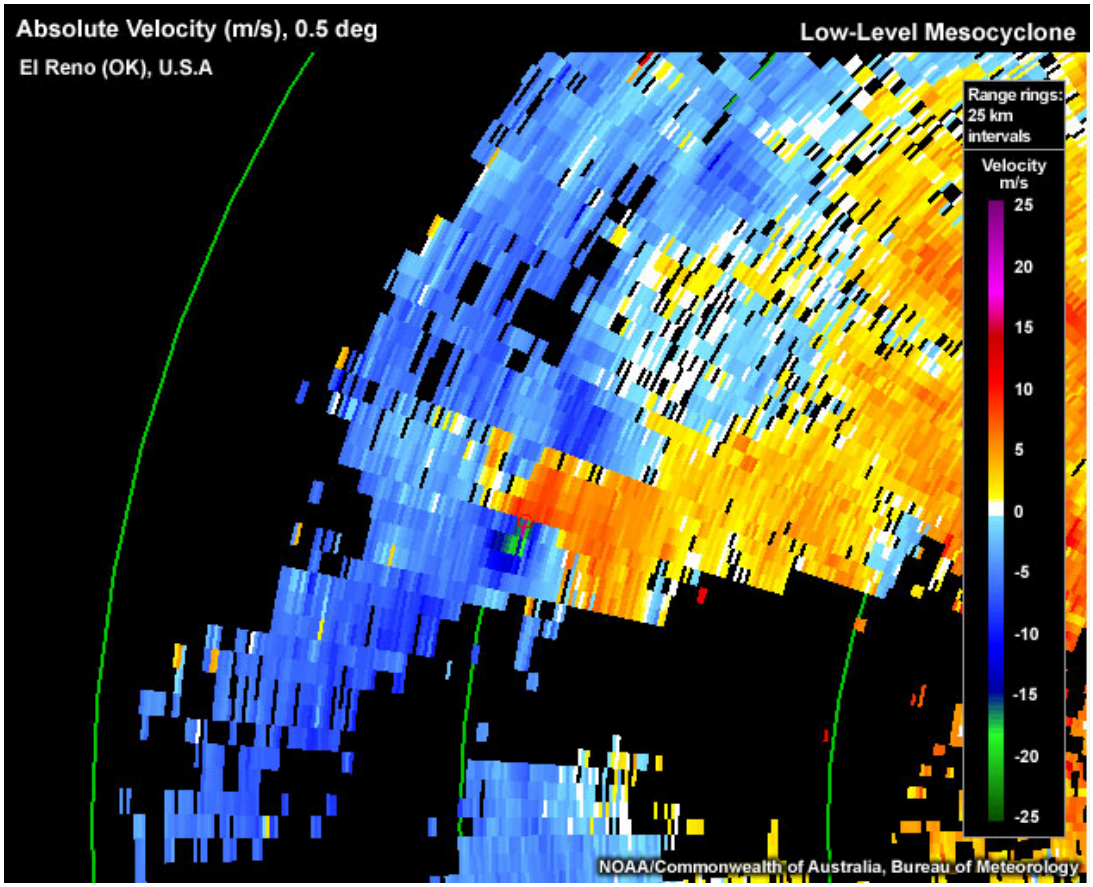

Figure 3.4: Example of rotation in radar-derived wind field. Rotation is indicated by a couplet (just below center of image) of strong inbound and outbound velocities along the same range ring (green line of constant distance from the radar). Image obtained from `http://www.meted.ucar.edu/radar/severe_signatures/index.htm`.

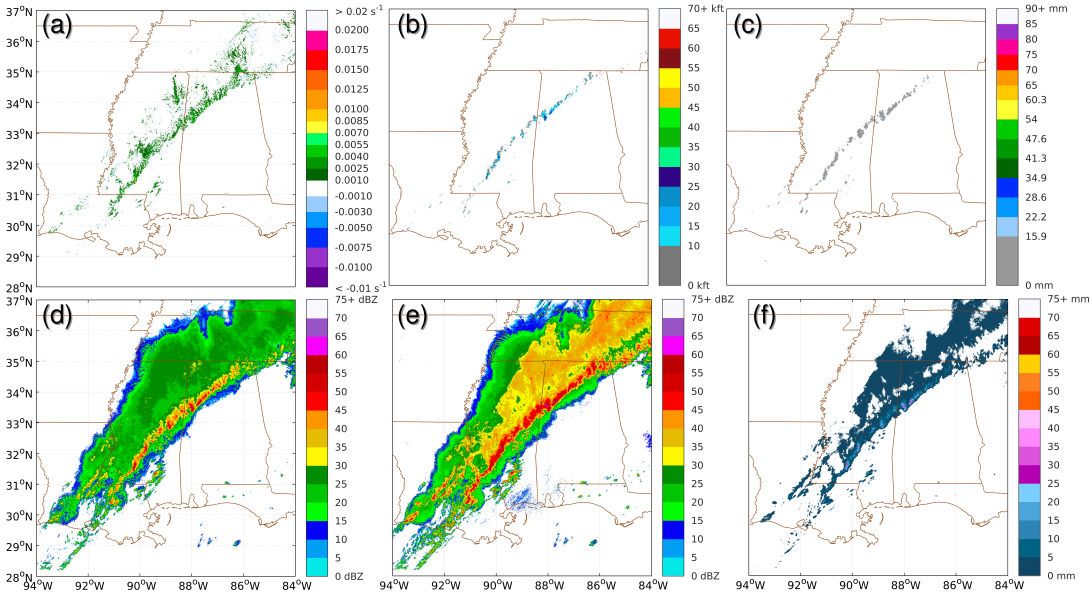

Figure 3.5: Sample radar images. (a) Low-level azimuthal shear; (b) 50-dBZ echo tops; (c) MESH; (d) -10 °C reflectivity; (e) composite reflectivity; (f) VIL.

reflectivity (Lakshmanan et al., 2006) from overlapping radars to obtain another 45 variables. Of these 45 variables, there are two types: single-level and composite. The single-level variables include reflectivity at 35 heights (0, 0.25, 0.5, 0.75, ... km above ground level) and three temperature levels (-20, -10, and 0 °C). The composite variables include all others listed in Table 3.2.

We do not use the 35 height-specific reflectivity fields, because (a) we assume that most of this information is subsumed by the isothermal and composite variables; (b) this would add 770 predictors to our training and testing data (22 per radar variable, as explained in Section 3.5), which would greatly increase computational requirements. Figure 3.5 shows examples of six radar variables in Table 3.2.

Variables	**Stats Calculated for Each**
Low-level (0–2-km) azimuthal shear	0^{th} percentile (minimum)
Mid-level (3–6-km) azimuthal shear	5^{th} percentile
18-dBZ echo top	25^{th} percentile (1^{st} quartile)
50-dBZ echo top	50^{th} percentile (median)
Max estimated hail size (MESH)	75^{th} percentile (3^{rd} quartile)
-20 °C reflectivity	95^{th} percentile
-10 °C reflectivity	100^{th} percentile (maximum)
0 °C reflectivity	Mean (1^{st} moment)
Composite (column-max) reflectivity	Standard deviation (related to 2^{nd} moment)
Lowest-altitude reflectivity	Skewness (related to 3^{rd} moment)
Severe-hail index (SHI)	Kurtosis (related to 4^{th} moment)
Vertically integrated liquid (VIL)	

Table 3.2: Radar statistics. For each variable, each statistic is calculated for all pixels inside the storm object. This is done for both raw values and gradient magnitudes.

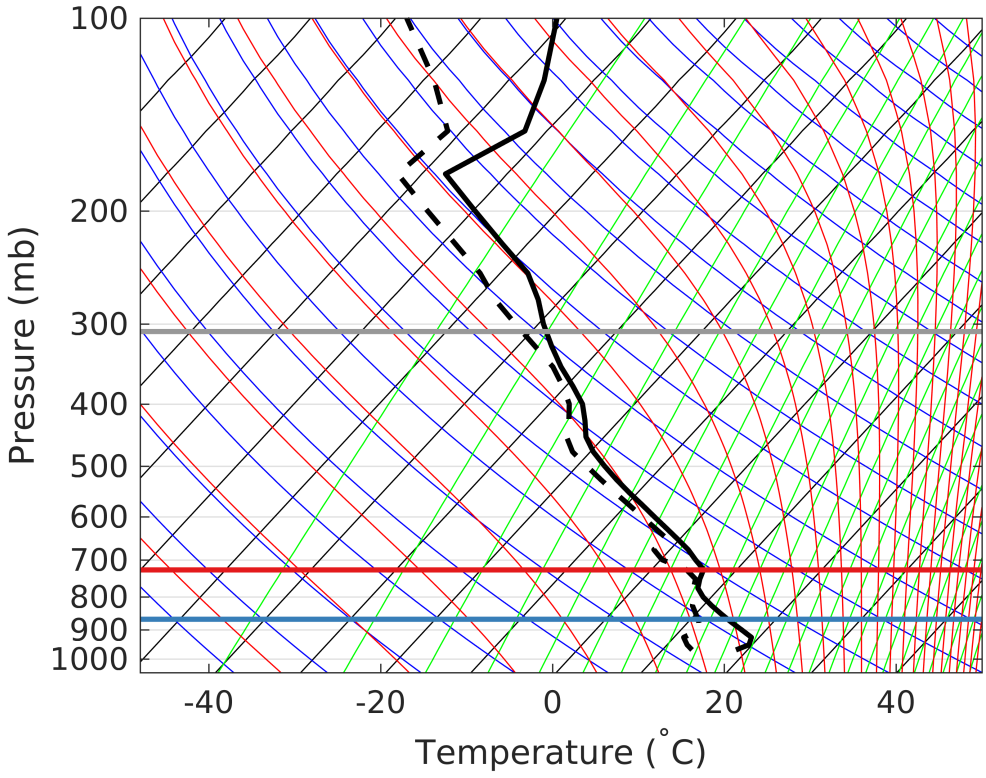

Figure 3.6: Sample sounding. The blue line is the lifting condensation level; red line is the level of free convection; and grey line is the equilibrium level, all for the most unstable parcel.

3.1.2 Model Soundings

A sounding is a vertical profile of five atmospheric variables: temperature, humidity, pressure, zonal wind speed (u-wind), and meridional wind speed (v-wind). See Figure 3.6. Soundings are used to calculate properties of the near-storm environment (NSE), which is important for determining how storm cells evolve. We use modeled rather than observed soundings, because observed soundings are very sparse (less than 100 sites in the CONUS, launching every 12 hours).

The Rapid Refresh (RAP) model is widely used to create other NSE datasets, such as the Storm Prediction Center's mesoanalysis[4]. However, the RAP was not available until May 2012, and our training/testing period ends in 2011 (see Section 3.2). Therefore, we use the RUC model, which is the RAP's predecessor. We use only zero-hour analyses (rather than true forecasts), because only zero-hour analyses are consistently available in the archive for the training/testing period.

The NARR is our backup model, used only when RUC data are not available. Since the NARR is a reanalysis (as opposed to a real-time forecasting model like the RUC), its archive is consistent and complete, which makes it ideal for this purpose. Again we use only zero-hour analyses, (a) to maintain consistency with the RUC data and (b) because the NARR does not provide true forecasts. We could have used many other reanalyses, but the NARR was the most convenient choice, because we had worked with it extensively in the past. We consider this a minor decision, since RUC data are missing for only 103 of 20,019 hours needed for the training/testing period.

3.1.3 Near-surface Wind Observations

At the outset of this project, we considered using remotely sensed data. For remotely sensed data to be viable, we would need a fine-resolution, high-quality, and wide-coverage (*e.g.,* several years and a significant portion of the CONUS) dataset with near-surface wind speeds. To our knowledge, the two closest facsimiles are radar-derived wind speeds (determined from radial velocities at overlapping radars, as discussed in Section 3.1.1) and synthetic-aperture radar (SAR) imagery, which is obtained by satellites. The main problem with radar-derived wind speeds is that radars measure at varying heights, usually far above the

[4] `http://www.spc.noaa.gov/exper/mesoanalysis`

standard measurement height of 10 m. The main problem with SAR is that its wind-speed-retrieval method works only above large water bodies. Thus, we abandoned the idea of using remotely sensed data and decided to use only *in situ* observations.

To our knowledge, MADIS is the largest collection of *in situ* weather observations in the world. MADIS is split into "data streams," each with data from a different type of observing platform. We use all data streams with wind observations from surface weather stations: cooperative stations, the Climate Reference Network (CRN), the Historical Climate Network (HCN), five-minute METARs, hourly METARs, maritime observations, mesonet stations, the New England Pilot Project (NEPP), standard aviation observations (SAOs), and urbanet stations. Unfortunately, MADIS does not contain data from the Oklahoma Mesonet or one-minute METARs. Both datasets contain high-quality observations with fine temporal resolution (five and one minutes, respectively), so we merge these with the MADIS dataset.

Finally, due to our need for more severe cases, we merge NWS severe-wind reports with the first three datasets. The NWS defines severe wind as a gust $\geq$ 50 kt, so we assume that all reports coincide with a gust $\geq$ 50 kt. Figure 3.7 shows an example of wind observations overlain with radar data.

We assume that all wind observations in the four datasets are straight-line (non-tornadic). This assumption is justified below.

1. NWS storm reports are split into three categories: straight-line winds, tornadoes, and hail. Thus, if wind damage were associated with a tornado, presumably the event would be recorded as a tornado rather than straight-line wind. We do not include tornado reports in our dataset.

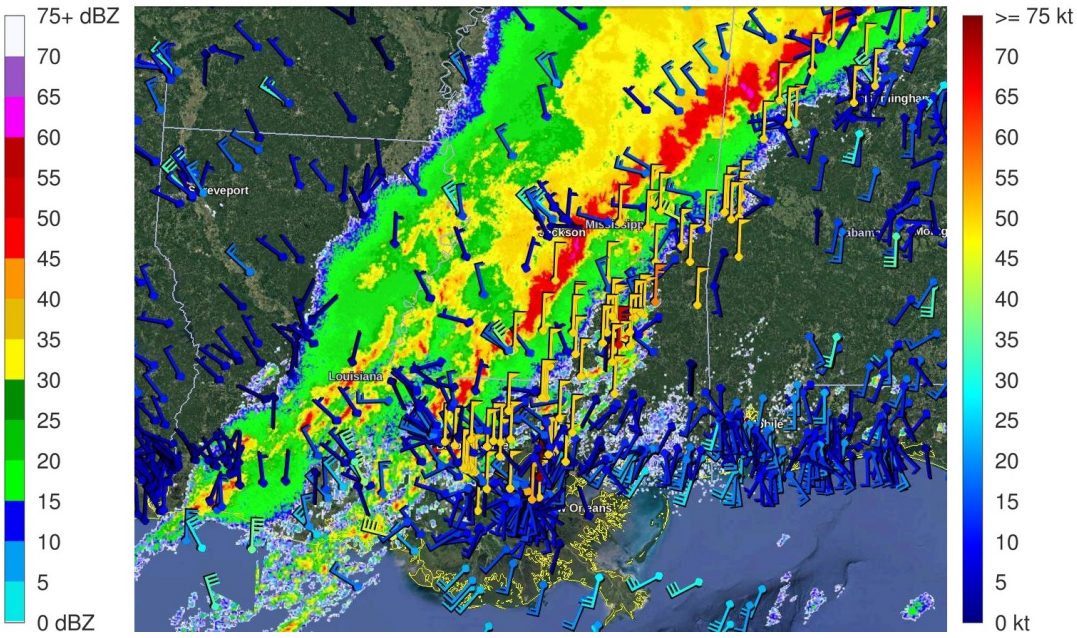

Figure 3.7: Wind observations overlain with radar data. The colour fill is composite reflectivity at 230011 UTC Apr 4 2011, and wind barbs show the maximum gust at each location over the next 90 minutes (230011 UTC Apr 4 . . . 003011 UTC Apr 5 2011). Wind gusts of $\geq$ 50 kt (orange and red) from due north are NWS reports. Since NWS reports do not contain a direction, these are plotted as due north by default.

2. Tornadoes are much less common[5] and more intense than damaging straight-line winds. Thus, very few tornadoes hit weather stations, and they are likely to destroy the anemometer if they do.

However, we accept that our dataset is probably contaminated by a small number of tornadic cases.

[5] `http://www.nssl.noaa.gov/education/svrwx101/wind`

3.2 Training and Testing Days

The datasets in Table 3.1 have a common period of July 2001 – December 2011, excluding 2009 (for which MYRORSS data were not available at processing time). However, we did not have enough computing power to process the ~3500 days in this period, so we eliminated the following:

1. Years 2001-03, for which MADIS and one-minute METARs (the two largest wind datasets) have many fewer stations.
2. Days with $<$ 30 NWS severe-wind reports in the CONUS. See Figure 3.8 for the cumulative density function (CDF) of daily severe-wind reports. 820 of 2557 days meet this criterion.
3. Days with $>$ 5 time steps of missing radar data (leads to missing predictor variables and broken storm tracks). 804 of 820 days satisfy this criterion, and we use these 804 days for all training and testing.

Using only days with $\geq$ 30 severe-wind reports (Figure 3.8) ensures a large number of both positive and negative cases (severe and non-severe-wind-producing storms, respectively). Clearly, on a day with many severe-wind reports, there should be many positive cases. However, there should also be many negative cases, because (a) severe-wind-producing storms are usually confined to a small area and time period; (b) even in the midst of a severe-wind outbreak, many storms are weak and do not produce severe winds.

3.3 Storm Detection and Tracking

Storm detection (the outlining of storm objects[6]) is done by `w2segmotionll` (Lakshmanan and Smith, 2010), which is an algorithm in the WDSS-II software

[6]Henceforth, I will use the terms "storm cell," "storm object," and "storm track". A "storm cell" will be a single thunderstorm (updraft-downdraft pair), consistent with common usage

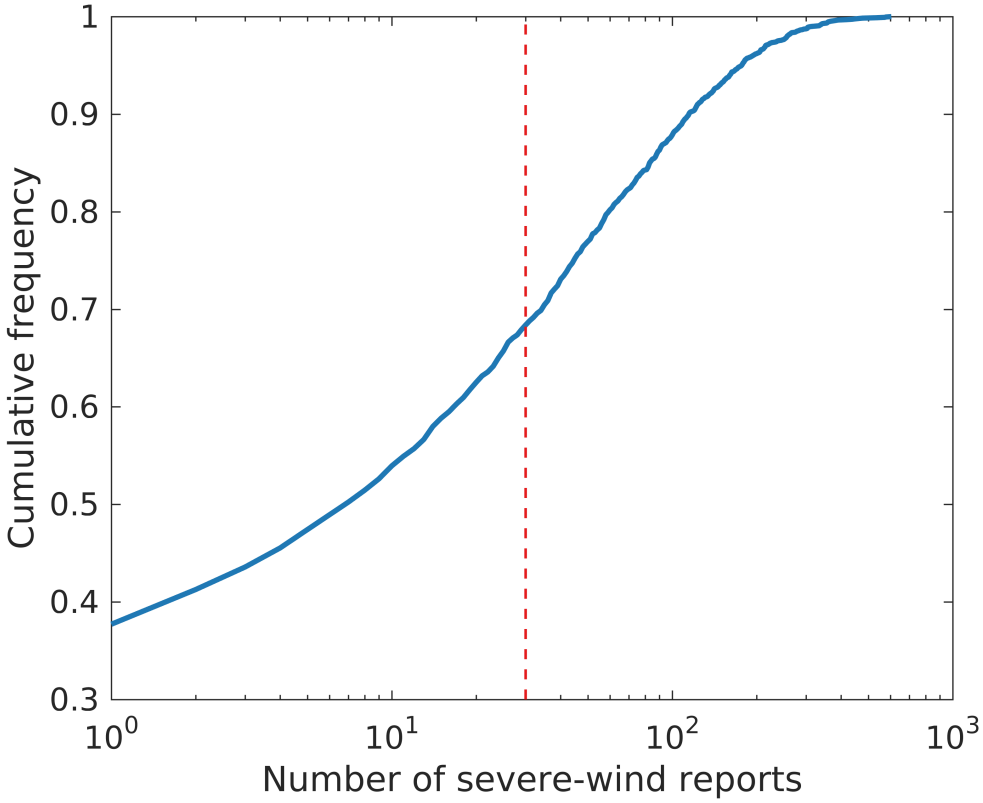

Figure 3.8: CDF of daily severe-wind reports for 2004-08 and 2010-11. Approximately 38% of days have no severe-wind reports, and approximately 30% reach the threshold (dashed red line) of 30 reports.

package (introduced in Section 3.1.1). The method used by `w2segmotionll` is called extended-watershed image segmentation (Lakshmanan et al., 2009) [an improvement over the original watershed method developed by Beucher and Lantuejoul (1979) and Beucher (1982)], which segments the radar image into locally maximum areas of the target variable (usually one of the reflectivity fields listed in Table 3.2).

Storm detection involves three input parameters: the target variable, minimum threshold for the target variable, and minimum storm-object area. We use -10 °C reflectivity with a minimum threshold of 30 dBZ, following Saxen (2002) and colleagues at the Cooperative Institute for Mesoscale Meteorological Studies (personal correspondence), who used these parameters successfully for lightning prediction. We use an area threshold of 50 km^2, based on subjective map analysis in which we found that this produces the best outline of the reflectivity core. For example, Figure 3.9 shows storm objects with a minimum area threshold of 25, 50, and 75 km^2. The 25-km^2 threshold leads to very small objects, which do not appear to be organized storm cells, and most of these are filtered out by the 50-km^2 threshold. Meanwhile, the 75-km^2 threshold leads to an oddly shaped storm cell (just east of 92° W, between 30° 36' and 30° 48' N) and filters out what appears to be a moderate-strength cell (maximum reflectivity of 45 dBZ, around 31° 24' N and 90° 30' W). Based on this and similar maps, we decided that 50 km^2 is the best compromise.

in meteorology. A "storm object" will be one storm cell at one time step, and a "storm track" will be a succession of storm objects linked in time (different snapshots of the same storm cell).

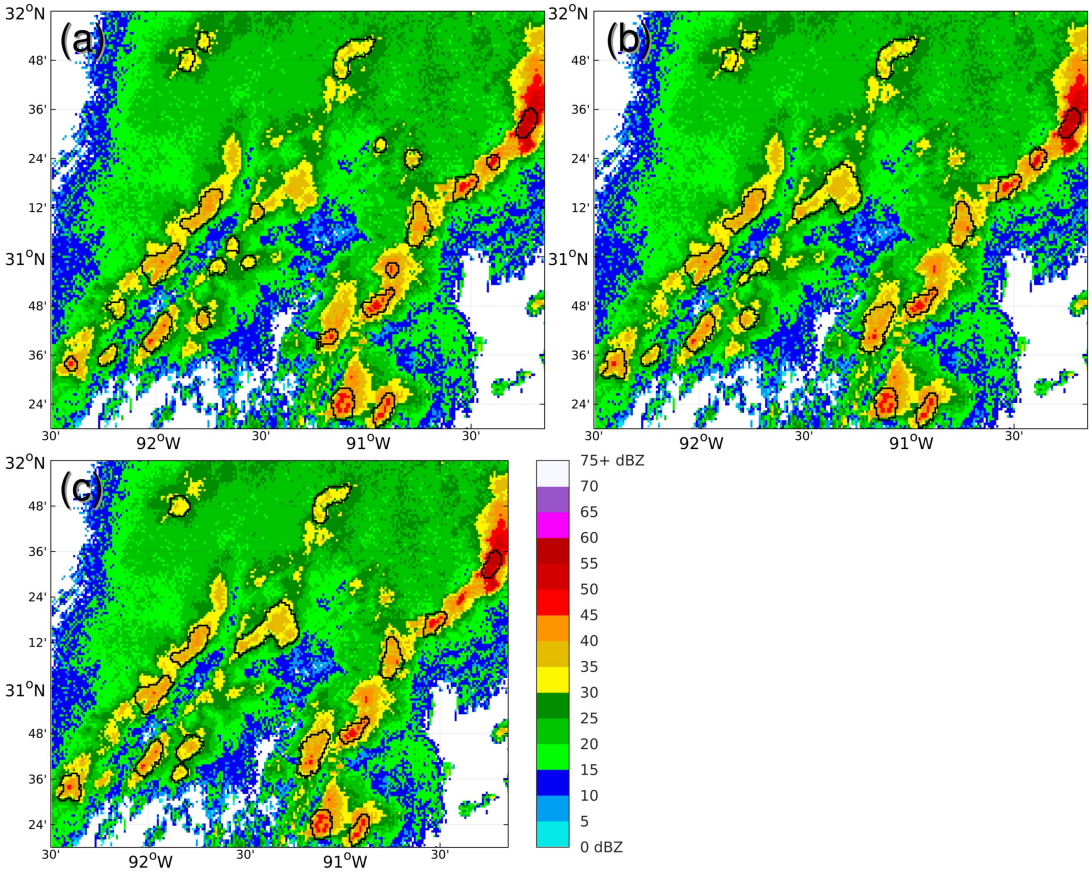

Figure 3.9: Comparison of minimum area thresholds for `w2segmotionll`: (a) 25, (b) 50, and (c) 75 km^2. The colour fill is -10 °C reflectivity, and the black outlines are detected storm objects.

Storm-tracking is done in two phases: real-time and post-event. Real-time tracking is done by `w2segmotionll`, and post-event tracking is done by a MATLAB implementation of `w2besttrack` (Lakshmanan et al., 2015), which is another WDSS-II algorithm[7]. The main difference is that, in order to connect a storm object with a track, `w2segmotionll` (and real-time algorithms in general) can use only information from the past, whereas `w2besttrack` (and post-event algorithms in general) can use information from both the past and future. This difference is illustrated in Figure 3.10.

The two algorithms are run sequentially. `w2segmotionll` creates preliminary storm tracks, which are updated by `w2besttrack`. The main advantage of `w2besttrack` is that it merges "broken" (falsely truncated) storm tracks, resulting in longer tracks on average. This allows storm objects to be linked with wind observations at greater lead times (Section 3.4), which allows machine-learning models to predict severe wind at greater lead times.

For example, suppose that a storm track lasts 90 minutes (1800-1930 UTC) according to `w2besttrack` but only 15 minutes (1800-1815 UTC) according to `w2segmotionll`. Also, suppose that a severe wind gust is linked (by the procedure in Section 3.4) to the storm object at 1930 UTC (S_{1930}). Using `w2besttrack`, this gust would also be linked to the storm object at 1800 UTC (S_{1800}), because `w2besttrack` knows that S_{1800} and S_{1930} are part of the same track (different snapshots of the same storm cell). However, using `w2segmotionll`, the wind gust would not be linked to S_{1800}, because S_{1800} and S_{1930} are not considered part of the same track. In other words, `w2besttrack` allows wind observations to be linked to storm cells with a greater lead time, which allows machine-learning models to make predictions with a greater lead time.

[7]Differences between the original WDSS-II code and MATLAB code are discussed at the end of this section.

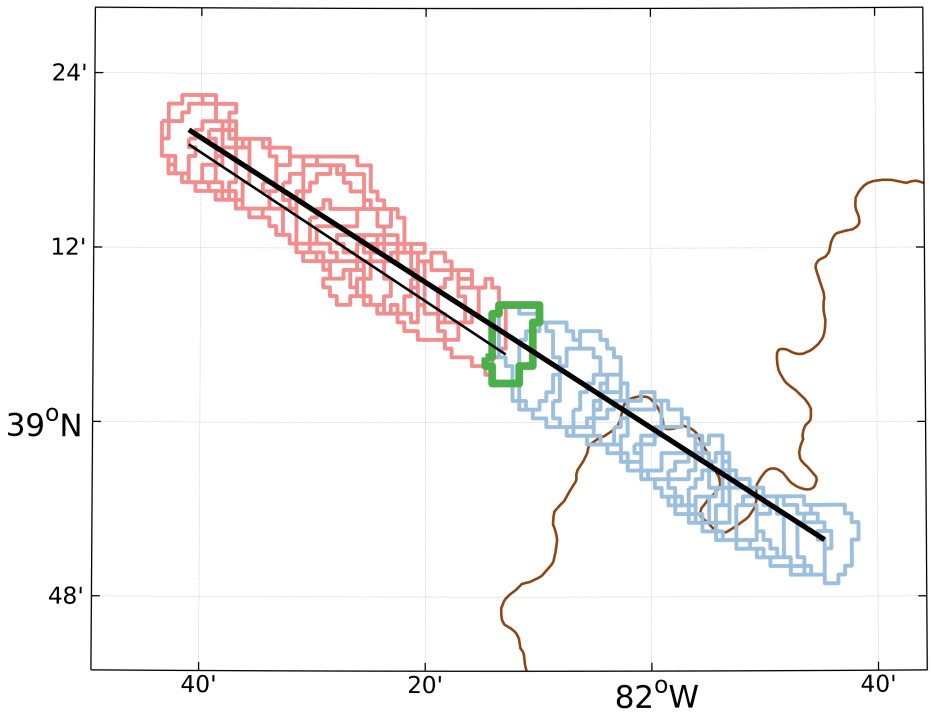

Figure 3.10: Comparison of storm-tracking procedures. When assigning the storm object at time t_0 (green) to a track, `w2besttrack` (thick black line) looks at storm objects in both the past (red) and future (blue). `w2segmotionll` (thin black line) looks only at storm objects in the past.

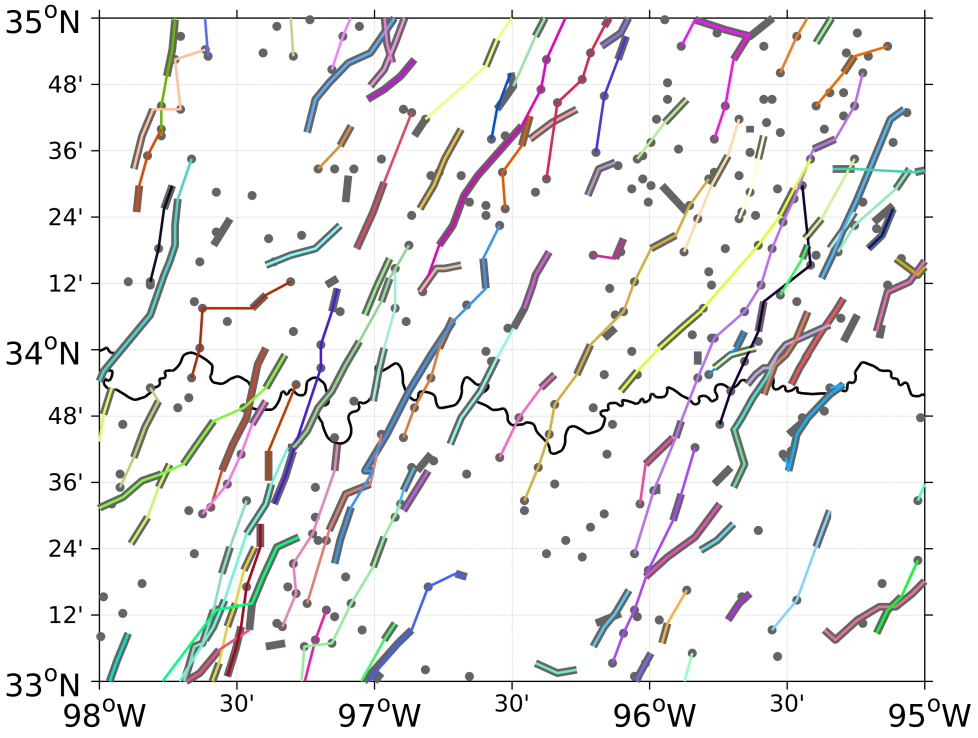

Figure 3.11: Comparison of storm-tracking results for a 24-hour period. Thick grey lines are tracks created by `w2segmotionll`; thin multi-coloured lines are tracks created by `w2besttrack`.

The main difference between the original and MATLAB versions of `w2besttrack` is that the MATLAB version is more memory-efficient[8], which allows more data (*i.e.*, more preliminary tracks from `w2segmotionll`) to be processed at the same time. This is important, because the training/testing period (Section 3.2) contains many sets of consecutive days, which must be processed at the same time. If consecutive days were not processed at the same time, storm tracks would be falsely truncated at the end of each day[9].

However, tracks are still falsely truncated at the end of each consecutive period (set of consecutive days). The only way to fix this problem would be to run `w2segmotionll` and `w2besttrack` for 90 minutes (the maximum lead time) after each consecutive period. However, this would take weeks of computing time, and we did not identify the issue until recently. Instead, when labeling storm objects (to indicate which ones are responsible for damaging straight-line wind) (Section 3.6), we ignore objects near the end of a consecutive tracking period.

[8]Data are continually swapped between random-access memory (RAM) and disk. When processing storm tracks from day n, a one-day buffer (tracks from days n - 1 and n + 1) is stored in RAM. All other data, including intermediate results, are cleared from RAM and stored on disk. The one-day buffer is sufficient, because storm cells never last longer than 24 hours.

[9]Tracks would also be falsely truncated at the beginning of each day. However, for each storm object we predict only winds in the future (at positive lead times), not in the past. Also, we do not use any data from the storm cell's past to create predictors, except for the speed of storm motion (see Section 3.5). Thus, we are not concerned with false truncation at the beginning of the day.

3.4 Linking Wind Observations to Storm Cells

The goal of this procedure is to causally attribute wind observations to storm cells. The procedure is described below for each wind observation W and buffer distance d (0, 5, or 10 km).

1. If W does not coincide with a radar-scan time, interpolate storm objects along their respective tracks to the time of W (t_W). Storm objects are interpolated as a whole – *i.e.,* their bounding polygons are simply advected, so their shapes do not change. Storm objects are interpolated from the nearest radar-scan time, so that interpolated polygons are most similar to the actual polygons that existed at t_W. This is important, because wind observations are linked to the nearest polygon edge, not to the nearest centroid.
2. Find the storm object with the nearest polygon edge. Let this object be S and the corresponding track be S^*.
3. If the nearest edge of S is within buffer distance d, link W to all storm objects in track S^*. Otherwise, do not link W at all.

Sample output is shown in Figure 3.12. In step 3, for each wind observation linked to a storm cell, this information is shared across all storm objects in the track. Linkages created by the above procedure are exploited in Section 3.6, where a label is calculated for each storm object (either severe-wind-producing or not, over a given buffer distance and lead-time window).

3.5 Calculation of Predictors

Four types of predictors are calculated for each storm object: radar statistics, storm motion, shape parameters, and sounding parameters.

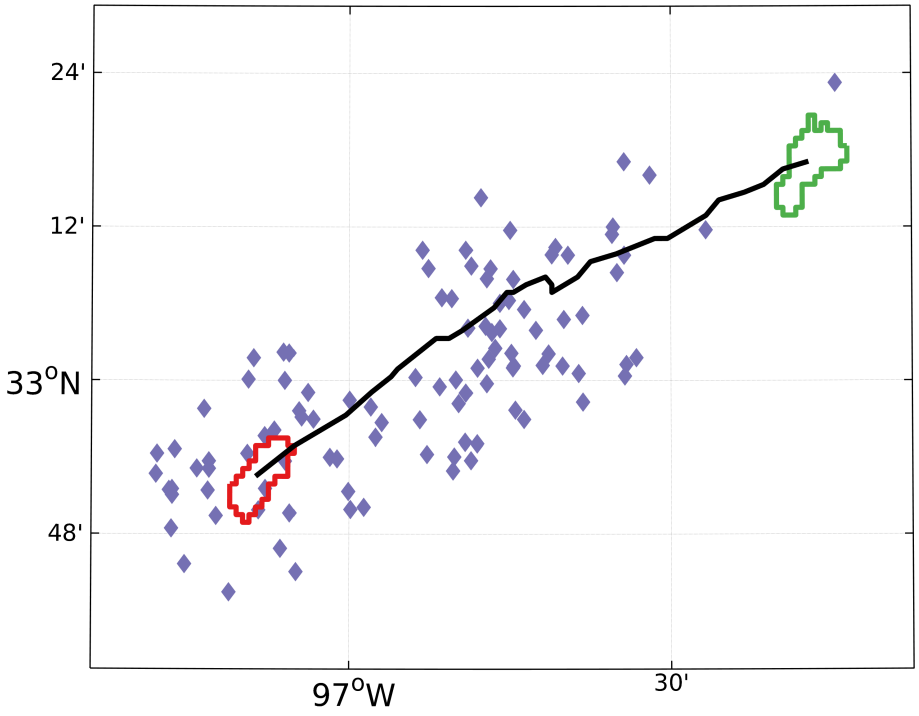

Figure 3.12: Sample output for linkage. The green polygon is the first, and red polygon is the last, storm object in the track. The black line is the track itself (trajectory formed by the centroids of all storm objects). The purple diamonds are wind observations linked to the storm cell at a buffer distance of 10 km.

First, 11 statistics are calculated for each of the 12 radar variables (see Table 3.2), using only values inside the storm object[10]. In other words, the storm object is used as a "cookie-cutter," to extract only the region of interest (Figure 3.13). Then the gradient field (consisting of first-order spatial differences) is calculated for each of the 12 variables, and the same 11 statistics are calculated for gradient magnitudes inside the storm object.

Second, storm motion (speed and direction) is computed from tracks produced by `w2besttrack`. For the first storm object in the track (S_1), motion is estimated by a first-order forward difference ($\frac{\text{position of } S_2 - \text{position of } S_1}{\text{time of } S_2 - \text{time of } S_1}$); for the last storm object in the track (S_{end}), motion is estimated by a first-order backward difference ($\frac{\text{position of } S_{end} - \text{position of } S_{end-1}}{\text{time of } S_{end} - \text{time of } S_{end-1}}$); for all other objects in the track (S_k), motion is estimated by a first-order centered difference ($\frac{\text{pstn of } S_{k+1} - \text{pstn of } S_{k-1}}{\text{time of } S_{k+1} - \text{time of } S_{k-1}}$).

Third, shape parameters are listed below.

1. Area: The areal extent (*e.g.,* 125 km^2) of the bounding polygon.
2. Orientation: The orientation of the best-fit ellipse for the bounding polygon. This angle is measured counterclockwise from the positive x-axis (due east) to the major axis and may vary from $[0, 180)^\circ$.
3. Eccentricity: The eccentricity of the best-fit ellipse.
4. Solidity: Proportion of pixels in the bounding polygon that are also in its convex hull.

[10]For low-level and mid-level azimuthal shear, values in the storm object are dilated, because strong rotation (*e.g.*, associated with mesocyclones) often occurs outside of the reflectivity core (which the storm object is meant to outline). Specifically, at each pixel (i, j), both the 10$^{\text{th}}$ and 90$^{\text{th}}$ percentiles (p_{10} and p_{90}) of azimuthal shear are taken from a 5-by-5 window centered on (i, j). Then the percentile with the greatest absolute value (either p_{10} or p_{90}) replaces the value at (i, j). Considering both p_{10} and p_{90} ensures that areas of both positive and negative shear are captured. This procedure is carried out independently for both low-level and mid-level azimuthal shear.

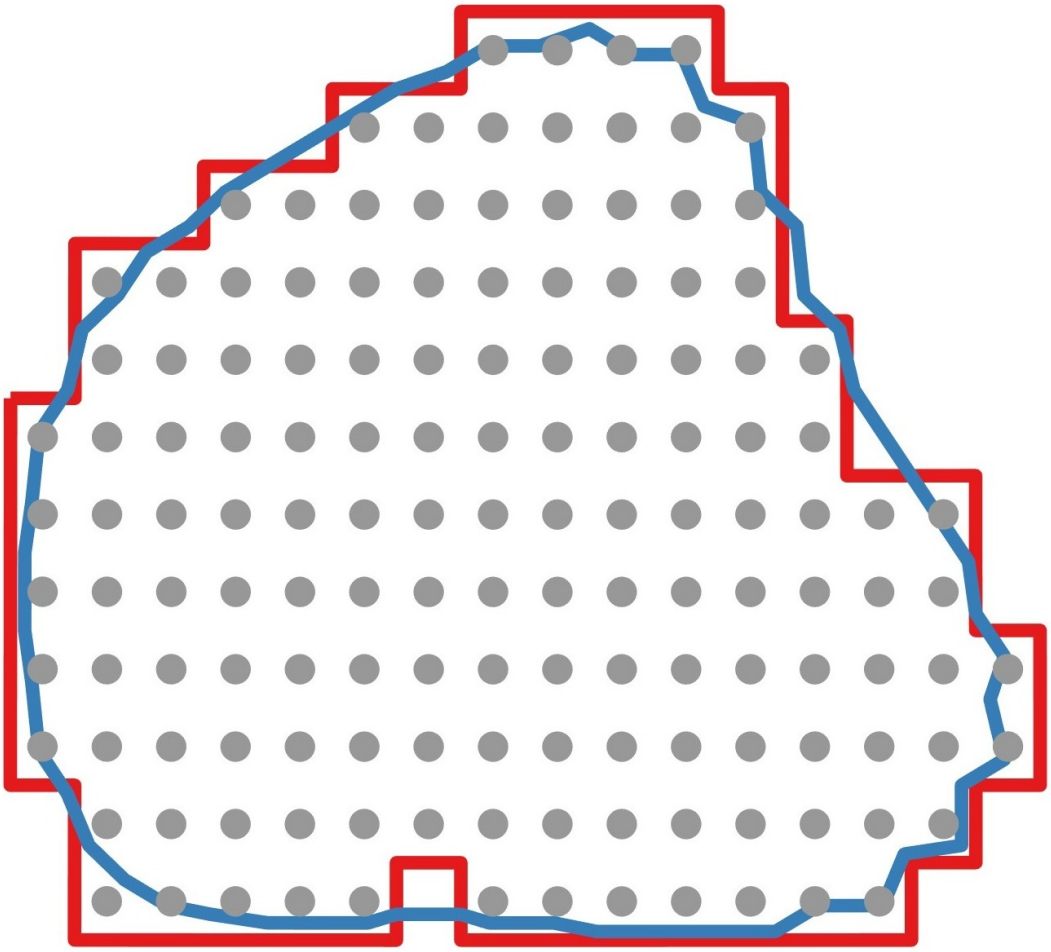

Figure 3.13: The red polygon is the raw, and blue polygon is the smoothed, version of the storm object. The grey dots are radar pixels inside the raw storm object. These pixels are used to calculate radar statistics.

5. Extent: Proportion of pixels in the bounding polygon that are also in its smallest bounding rectangle.
6. Curvature: Mean absolute curvature over all vertices of the bounding polygon.
7. Bending energy: Sum of squared curvatures (over all vertices of the bounding polygon), divided by perimeter.
8. Compactness: Squared perimeter, divided by 4π times the area.

Curvature, bending energy, and compactness are based on the smoothed polygon; all other parameters are based on the raw polygon (Figure 3.13). This is because 90° angles in the raw polygon (which is simply an outline of all radar pixels in the storm object) lead to unrealistically large curvature and perimeter values. We use smoothing via iterative averaging (SIA) (Mansouryar and Hedayati, 2012), with five iterations and a two-vertex radius of influence. We experimented with other values and found (subjectively) that these are the smallest values required to produce a realistic (not obviously pixelated) polygon. Thus, any further smoothing would have been unnecessary and might have removed important shape characteristics.

Fourth, the RUC sounding is interpolated to the time and centroid of the storm object. Specifically, the following variables are interpolated independently at each pressure level (Table 3.3): temperature, relative humidity, geopotential height, u-wind, and v-wind. The near-surface version of each variable is also interpolated. Linear temporal interpolation is done first (to the time of the radar scan), and then bicubic spatial interpolation is done (to the centroid of the storm object at said time). We would have preferred a more sophisticated temporal method, such as cubic splines, but this increased computing time for interpolation approximately tenfold, which was unacceptable.

Model	Pressure Levels	Variables at Each Pressure Level	Near-surface Variables
RUC	[100, 1000] mb	Temperature	2-m temperature
	in 25-mb intervals	Relative humidity	2-m relative humidity
		Geopotential height	Surface pressure
		u-wind	10-m u-wind
		v-wind	10-m v-wind
NARR	[100, 300] mb	Temperature	2-m temperature
	in 25-mb intervals	Specific humidity	2-m specific humidity
	[350, 700] mb	Geopotential height	Surface pressure
	in 50-mb intervals	u-wind	10-m u-wind
	[725, 1000] mb	v-wind	10-m v-wind
	in 25-mb intervals		

Table 3.3: Models used to create soundings. 2-m and 10-m stand for 2-metre and 10-metre, respectively.

If RUC data are missing for either the previous or next hour, soundings are created with NARR data, instead. All interpolation methods are the same, except that the NARR has a different moisture variable (specific humidity) and different pressure levels (Table 3.3).

For each storm object, we use the SHARPpy software (Halbert et al., 2015) to calculate the 97 parameters listed in Table 3.4. For all parameters involving storm motion [except those based on a "left-mover" or "right-mover," which use Bunkers et al. (2000) motion estimates], we use the first-order motion estimate calculated from `w2besttrack`, as described above.

All vector quantities (*e.g.,* storm motion, wind vectors, wind-shear vectors) are decomposed into magnitude, sine, and cosine, so that values scale linearly

for learning. Decomposing into magnitude and direction would have introduced a circular variable, which most learning algorithms are unable to handle. Also, decomposing into the x- and y-components would have de-emphasized the importance of the overall magnitude. The only exception to this procedure is for orientation. Although orientation is a vector quantity, it is a unit vector (simply pointing in a certain direction), which means that its magnitude is always 1.0. Thus, we decompose orientation into only sine and cosine. After decomposing vectors, there are 431 predictor variables: 264 radar statistics; magnitude, sine, and cosine of the motion vector; 9 shape parameters; and 155 sounding parameters.

We also considered using temporal changes of some predictors to create new predictors. For the original predictor x_j, the temporal predictor would be a first-order backward difference. For a storm object at time t_k, this would be $\frac{(x_j \text{ for } S \text{ at } t_k) - (x_j \text{ for } S \text{ at } t_{k-1})}{t_k - t_{k-1}}$. However, we decided against this, because our models are designed for use in a real-time forecasting environment (Section 6.3). In a real-time environment, only real-time storm tracks (from `w2segmotionll`) are available, which tend to be very short (Figure 3.11). Thus, many storm objects (S at t_k) are the first in their respective tracks, which means that they have no predecessor (S at t_{k-1}). Temporal variables could not be calculated for these objects.

Variable	**Description of Variable**	**Units**	**Vector?**
`brn`	Bulk Richardson number (BRN)	None	
`brnDenom`	BRN denominator	$m^2\ s^{-2}$	
`brnShear`	BRN shear term	$m\ s^{-1}$	Yes
`capStrength`	Cap strength	K	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
cape	Convective available potential energy (CAPE)	J kg^{-1}	
cape3km	CAPE from 0-3 km above ground level (AGL)	J kg^{-1}	
cape6km	CAPE from 0-6 km AGL	J kg^{-1}	
capeFreezing	CAPE from surface – freezing level	J kg^{-1}	
cin	Convective inhibition	J kg^{-1}	
convectiveTemp	Convective temperature	K	
critAngle	Critical angle	°	
crossTotals	Cross-totals index	K	
dcape	Downdraft CAPE	J kg^{-1}	
dcp	Derecho composite parameter	None	
effBwd	Wind difference over effective bulk layer (EBL)	m s^{-1}	Yes
effLayerBottom	Effective-layer bottom	m	
effLayerDepth	Effective-layer depth	m	
effLayerTop	Effective-layer top	m	
effShear	Effective-layer shear	m s^{-1}	Yes
ehi1km	Energy helicity index (EHI) from 0-1 km AGL	J kg^{-1}	
ehi3km	EHI from 0-3 km AGL	J kg^{-1}	
ehiLeft	Effective-layer EHI for left-mover	J kg^{-1}	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
ehiRight	Effective-layer EHI for right-mover	J kg^{-1}	
elHeight	Height AGL of equilibrium level	m	
esp	Enhanced stretching potential	None	
fosberg	Fosberg fire-weather index	None	
height0C	Height AGL of 0 °C isotherm	m	
height-10C	Height AGL of -10 °C isotherm	m	
height-20C	Height AGL of -20 °C isotherm	m	
height-30C	Height AGL of -30 °C isotherm	m	
kIndex	K-index	K	
lapseRate3km	Lapse rate from 0-3 km AGL	K km^{-1}	
lapseRate3-6km	Lapse rate from 3-6 km AGL	K km^{-1}	
lapseRate700-500mb	Lapse rate from 700-500 mb	K km^{-1}	
lapseRate850-500mb	Lapse rate from 850-500 mb	K km^{-1}	
lclHeight	Height AGL of lifting condensation level	m	
lfcHeight	Height AGL of level of free convection	m	
lhp	Large-hail parameter	None	
li300mb	Lifted index from surface – 300 mb	K	
li500mb	Lifted index from surface – 500 mb	K	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
liMax	Max lifted index in column (surface – any level)	K	
maxWindPbl	Max wind in planetary boundary layer (PBL)	m s^{-1}	Yes
mburst	Microburst index	None	
meanEffBulkWind	Mean EBL wind	m s^{-1}	Yes
meanEffWind	Mean effective-layer wind	m s^{-1}	Yes
meanMixr100mb	Mean mixing ratio in first 100 mb AGL	g kg^{-1}	
meanRh1km	Mean relative humidity (RH) from 0-1 km AGL	%	
meanRh150mb	Mean RH from 0-150 mb AGL	%	
meanRh150-350mb	Mean RH from 150-350 mb AGL	%	
meanRhPbl	Mean RH in PBL	%	
meanWind1km	Mean wind from 0-1 km AGL	m s^{-1}	Yes
meanWind3km	Mean wind from 0-3 km AGL	m s^{-1}	Yes
meanWind6km	Mean wind from 0-6 km AGL	m s^{-1}	Yes
meanWind8km	Mean wind from 0-8 km AGL	m s^{-1}	Yes
meanWindLclEl	Mean wind from lifting condensation level – equilibrium level (LCL-EL)	m s^{-1}	Yes
meanWindPbl	Mean wind in PBL	m s^{-1}	Yes
minBuoyancy	Minimum buoyancy in column	K	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
mmp	Mesoscale convective system (MCS) maintenance probability	%	
mplHeight	Height AGL of max parcel level	m	
pblDepth	Depth of PBL	m	
pw	Precipitable water	mm	
rhSurface	Surface RH	%	
scpLeft	Supercell composite parameter (SCP) for left-mover	None	
scpRight	SCP for right-mover	None	
shear1km	Wind shear from 0-1 km AGL	m s^{-1}	Yes
shear3km	Wind shear from 0-3 km AGL	m s^{-1}	Yes
shear6km	Wind shear from 0-6 km AGL	m s^{-1}	Yes
shear8km	Wind shear from 0-8 km AGL	m s^{-1}	Yes
shear9km	Wind shear from 0-9 km AGL	m s^{-1}	Yes
shearLclEl	Wind shear from LCL-EL	m s^{-1}	Yes
sherb	Severe hazards in environments with reduced buoyancy (SHERB) parameter	None	
ship	Significant-hail parameter	None	
sigSevere	Significant-severe parameter	None	
srh1km	Storm-relative helicity (SRH) from 0-1 km AGL	J kg^{-1}	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
srh3km	SRH from 0-3 km AGL	J kg^{-1}	
srhLeft	Effective-layer SRH for left-mover	J kg^{-1}	
srhRight	Effective-layer SRH for right-mover	J kg^{-1}	
srw1km	Storm-relative wind (SRW) from 0-1 km AGL	m s^{-1}	Yes
srw2km	SRW from 0-2 km AGL	m s^{-1}	Yes
srw3km	SRW from 0-3 km AGL	m s^{-1}	Yes
srw4-5km	SRW from 4-5 km AGL	m s^{-1}	Yes
srw4-6km	SRW from 4-6 km AGL	m s^{-1}	Yes
srw6km	SRW from 0-6 km AGL	m s^{-1}	Yes
srw8km	SRW from 0-8 km AGL	m s^{-1}	Yes
srw9-11km	SRW from 9-11 km AGL	m s^{-1}	Yes
srwBulk	Mean SRW in EBL	m s^{-1}	Yes
srwEff	Mean effective-layer SRW	m s^{-1}	Yes
srwLclEl	Mean SRW from LCL-EL	m s^{-1}	Yes
stpEff	Significant-tornado parameter (STP) for effective layer	None	
stpFixed	STP for fixed layer	None	
sweat	SWEAT index	None	
thetaeDiff	Difference between min and max equivalent potential temperature (θ_e) from 0-3 km AGL	K	

Continued on next page

Table 3.4 – continued from previous page

Variable	Description of Variable	Units	Vector?
`thetaeIndex`	θ_e-index	K	
`totalTotals`	Total-totals index	K	
`updraftTilt`	Updraft tilt	°	
`verticalTotals`	Vertical-totals index	K	
`wdi`	Wind-damage index	None	

Table 3.4: Sounding parameters.

3.6 Calculation of Labels

The storm object is labeled for each buffer distance (0, 5, and 10 km) and lead-time window ($[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes). This label (either 0 or 1) indicates whether or not the storm is responsible for damaging straight-line wind and is used as the predictand (dependent variable) for machine learning.

The labeling procedure is described by Algorithm 3.1, where the "end of consecutive period" criterion ensures that labels are not affected by breaks in storm-tracking (see Section 3.3). Sample output is shown in Figure 3.14.

3.7 Choice of Buffer Distances and Lead Times

This section justifies the choice to make separate forecasts for each buffer distance and lead-time window. Primarily, we assume that predictor-predictand

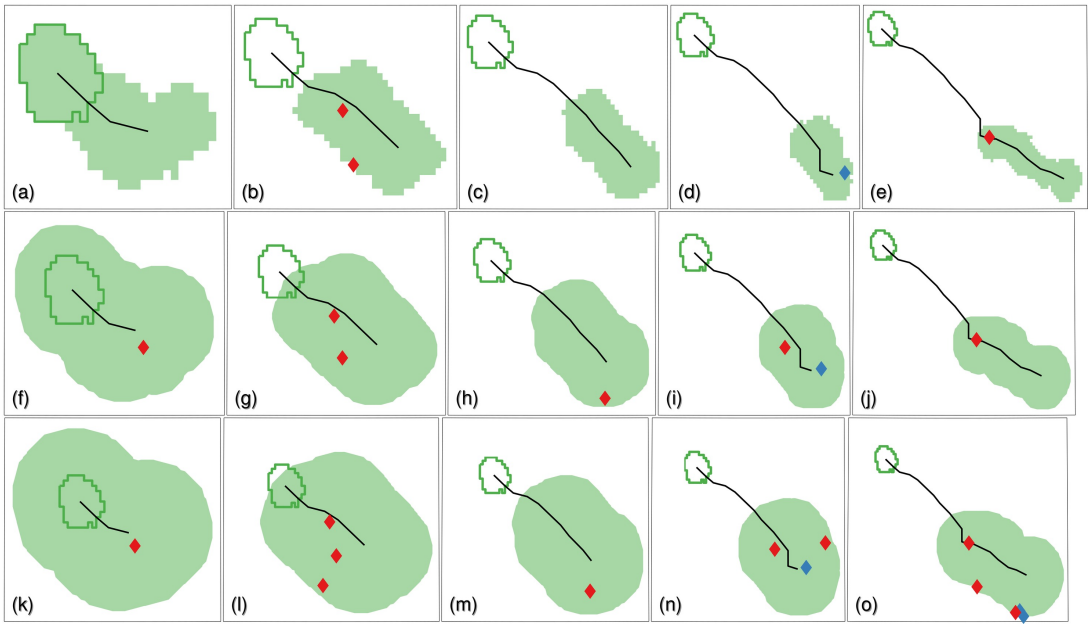

Figure 3.14: Labeling of a storm object for each buffer distance and lead-time window. Buffer distance increases from top to bottom: 0 km in panels (a)-(e), 5 km in (f)-(j), and 10 km in (k)-(o). Lead time increases from left to right: $[0, 15]$ minutes in panels (a), (f), (k); $[15, 30]$ minutes in panels (b), (g), (l); $[30, 45]$ minutes in panels (c), (h), (m); $[45, 60]$ minutes in panels (d), (i), (n); and $[60, 90]$ minutes in panels (e), (j), (o). The dark green polygon is the storm object at time t_0, which is to be labeled. The light green fill is the buffered area covered by distance d around all storm objects in the same track from times $[t_0 + \Delta t_{min}, t_0 + \Delta t_{max}]$, where Δt_{min} and Δt_{max} are the minimum and maximum lead times. Red diamonds are severe wind gusts ($\geq$ 50 kt); blue diamonds are non-severe. For each panel, if there is at least one red diamond in the buffered area, the storm object is labeled 1; otherwise, it is labeled 0.

Algorithm 3.1: `label_storm_object`

Input: d = buffer distance; $[\Delta t_{min}, \Delta t_{max}]$ = lead-time window; S = storm object

Output: `label`

if S is linked to wind gust $\geq$ 50 kt within buffer distance d and lead-time window $[\Delta t_{min}, \Delta t_{max}]$ **then**
    `label` = 1;
else
    if S occurs within t_{max} of end of consecutive period **then**
        `label` = null;
    else
        `label` = 0;
    end
end

relationships change with buffer distance and lead time. For example, if a storm cell produces severe wind at $t_0 + 90$ minutes, it should have certain features at t_0 that foretell its longevity. If a storm cell produces severe wind at $t_0 + 5$ minutes, it need not have such features.

Also, note that the lead-time windows ($[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes) are disjoint but the spatial buffers are overlapping. As shown in Figure 3.14, the 0-km buffer includes all points inside the storm object; the 5-km buffer includes all points inside and within 5 km of the storm object; and the 10-km buffer includes all points inside and within 10 km of the storm object. In other words, the 5-km buffer includes the 0-km buffer and the 10-km buffer includes the 5-km buffer. In order to fully separate the predictor-predictand relationships at different spatial scales, these buffers should have been made disjoint, as shown in Figure 3.15.

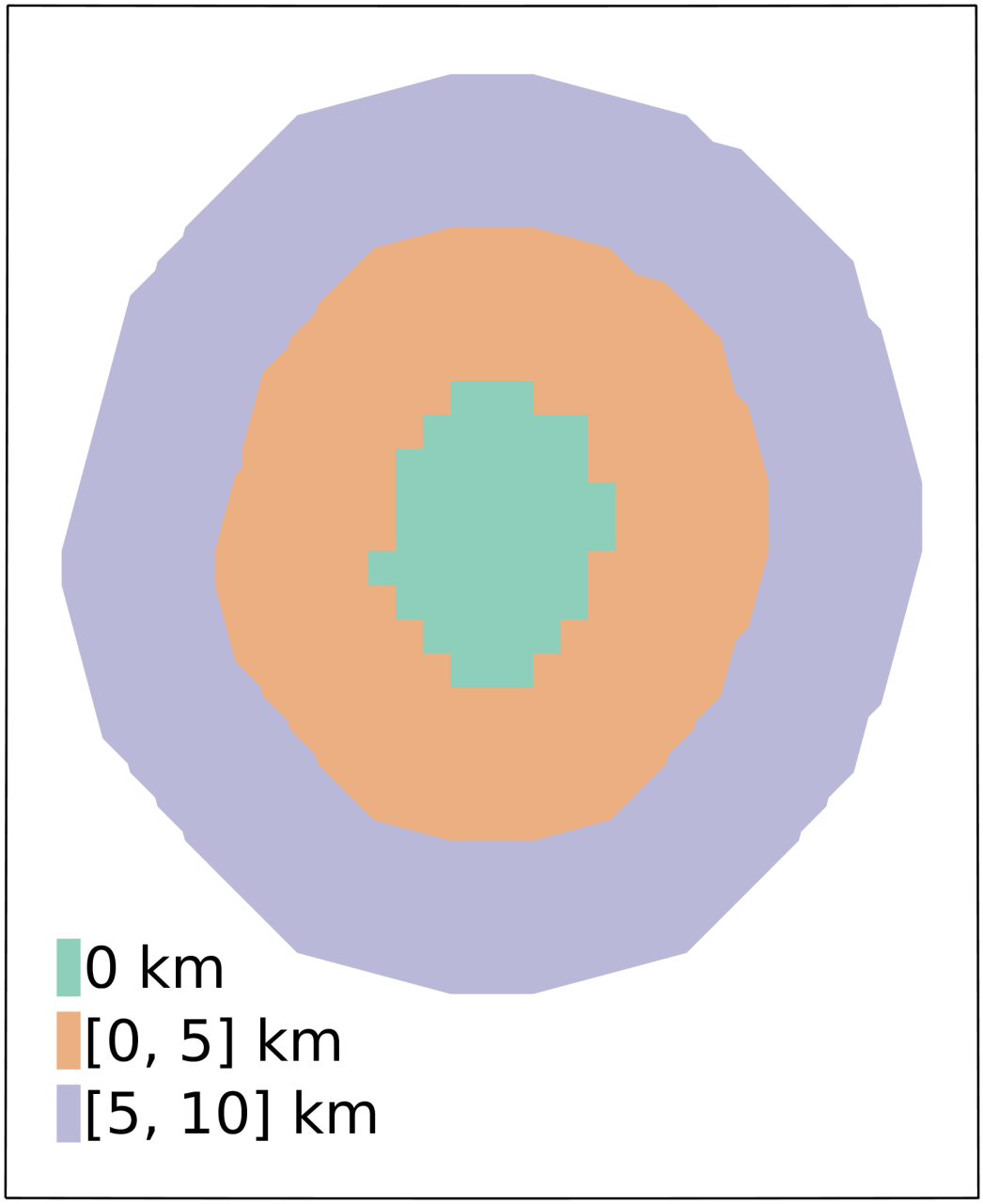

Figure 3.15: Disjoint spatial buffers around a storm object.

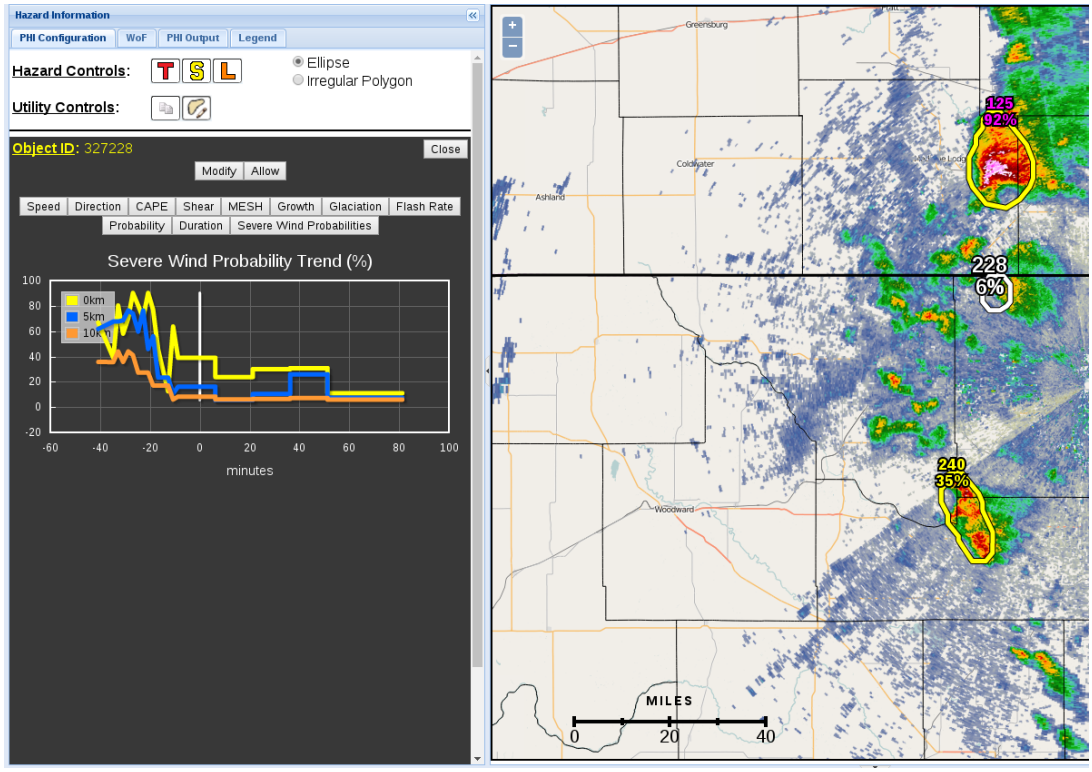

Figure 3.16: Sample forecast (graph at left) for one storm object from the 2016 SFE.

However, for the Spring Forecasting Experiment, forecasts were shown in the Probabilistic Hazard Information (PHI) tool (described in Section 6.3). The PHI tool can show probability-time graphs (*e.g.*, Figure 3.16), but it cannot display spatial buffers around the storm object. In other words, PHI allows temporal forecasts to be composited in a visually intuitive way, but it does not do the same for spatial forecasts. Thus, if we had used disjoint spatial buffers (Figure 3.15), users would have been forced to imagine polygons with holes – rather than filled polygons, which are much easier to visualize.

Chapter 4

Machine-learning Methods

4.1 Algorithms

This section will discuss two types of algorithms: those used to create the base model and those used for probability calibration. Probability calibration is necessary because (a) most base models generate poorly calibrated probabilities (Niculescu-Mizil and Caruana, 2005) and (b) the base model is trained on a uniform distribution of maximum storm-object winds, which is very unlike the population distribution. For more on sampling techniques, see Section 4.2.

4.1.1 Base Models

4.1.1.1 Logistic Regression

Logistic regression (LR) (Walker and Duncan, 1967) is similar to linear regression, except that it fits a logit curve to the training data, rather than a straight line. The logit curve is described by the following equation (see Figure 4.1).

$$f_i = \frac{\exp(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})}{1 + \exp(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})} = \frac{e^{(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})}}{1 + e^{(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})}} \tag{4.1}$$

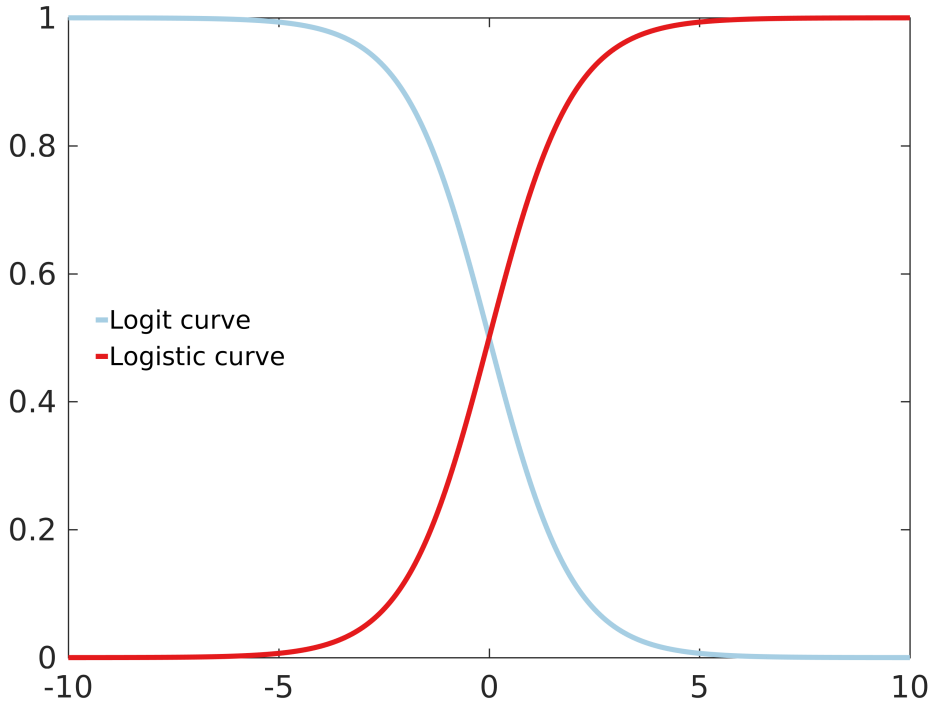

Figure 4.1: The logit curve is plotted as $\frac{e^{-x}}{1+e^{-x}}$. The logistic curve is plotted as $\frac{1}{1+e^{-x}}$.

f_i is the forecast probability of label = 1 for the i^{th} storm object; x_{ij} is the value of the j^{th} predictor for the i^{th} storm object; N is the number of predictors (431); β_0 is the bias term; and β_j is the coefficient for the j^{th} predictor. As the exponent $r \to -\infty$, $e^r \to 0$ and $f_i \to 0$. As the exponent $r \to +\infty$, $e^r \to \infty$ and $f_i \to 1$. Thus, f_i is bounded by $[0, 1]$, which is analogous to a true probability. This is why LR is so often used for binary classification.

During training, gradient descent (Section 4.4.3 of Mitchell, 1997) is used to minimize the model deviance. In other words, the model deviance, defined below, is the "objective function".

$$D = -\frac{1}{M}\sum_{i=1}^{M}[y_i \log_2(f_i) + (1 - y_i)\log_2(1 - f_i)] \tag{4.2}$$

y_i is the label for the i^{th} storm object; f_i has the same meaning as in Equation 4.1; and M is the number of storm objects in the training set.

Unfortunately, LR is not well suited for problems with a large number of predictor variables. This is because LR does not perform variable selection, so it usually includes many unimportant variables (*i.e.,* those that are weakly correlated with the predictand or redundant with other predictor variables). Thus, we use LR for only two purposes: (a) as a baseline method, against which to compare more sophisticated methods, and (b) as a base model for sequential forward selection (Section 5.1.1).

4.1.1.2 Logistic Regression with an Elastic Net

Logistic regression with an elastic net (Zou and Hastie, 2005) (LREN) is similar to basic LR, except that the objective function is different. LREN uses gradient descent to minimize the "penalty," defined below.

$$P = D + \lambda \sum_{j=1}^{N} \left[\frac{1}{2}(1-\alpha){\beta_j}^2 + \alpha|\beta_j| \right] \tag{4.3}$$

D, M, and β_j have the same meanings as in Equations 4.1 and 4.2. λ is the regularization parameter, and α is the elastic-net parameter. For greater λ-values, fewer non-zero coefficients are produced and fewer variables are admitted into the model. For smaller λ-values, the opposite is true. This allows LREN to perform variable selection, unlike basic LR. Meanwhile, for greater α-values, the L_1-penalty is weighted more heavily; for smaller α-values, the L_2-penalty is weighted more heavily. The L_1- and L_2-penalties are $\frac{1}{2}(1-\alpha){\beta_j}^2$ and $\alpha|\beta_j|$, respectively. In the limit of $\alpha = 1$, LREN simplifies to lasso regression; for $\alpha = 0$, LREN simplifies to ridge regression. Thus, an "elastic net" is simply a weighted combination of lasso and ridge regression.

LREN usually outperforms basic LR, because it eliminates unimportant variables.

4.1.1.3 Feed-forward Neural Nets

A feed-forward neural net (FFNN) (Haykin, 2001) consists of several layers of neurons, with connections between neurons in adjacent layers[1]. Usually the network is fully connected, which means that each neuron in the k^{th} is connected to all neurons in the $(k - 1)^{\text{th}}$ and $(k + 1)^{\text{th}}$ layers. The simplest FFNN has three layers: an input layer, hidden layer, and output layer. In general FFNNs can have multiple hidden layers, but there is almost always one input layer and one output layer. More hidden layers allow an FFNN to learn more complex relationships between the predictors and predictand. However, runtime and memory requirements increase sharply with the number of layers, so in practice most studies use only a few hidden layers.

Each neuron has three properties: its weighted connections to other neurons (both forward, towards the output layer, and backwards, towards the input layer); an input function, which transforms the neuron's inputs into a scalar; and an activation function, which transforms this scalar into an output. The input function is usually a linear transformation, defined below.

$$a_j{}^{(k+1)} = w_{0j}{}^{(k)} + \sum_{i=1}^{N^{(k)}} w_{ij}{}^{(k)} z_i{}^{(k)} \tag{4.4}$$

$a_j{}^{(k+1)}$ is the result of the input function, sometimes called the "activation," for the j^{th} neuron in the $(k + 1)^{\text{th}}$ layer; $w_{ij}{}^{(k)}$ is the weight for the connection from the i^{th} neuron in the k^{th} layer to the j^{th} neuron in the $(k + 1)^{\text{th}}$ layer; $z_i{}^{(k)}$ is the output from the i^{th} neuron in the k^{th} layer; and $N^{(k)}$ is the number of neurons in the k^{th} layer. For the input layer, change $z_i{}^{(k)}$ to x_i, the i^{th} predictor variable, and $N^{(k)}$ to $N = 431$, the number of predictor variables.

[1] A neural net may also have "shortcut connections," between neurons in non-adjacent layers, in which case it is called a cascade-forward neural net. See Figure 4.2. However, this greatly increases the number of connections, which greatly increases runtime and memory requirements, so we use only FFNN.

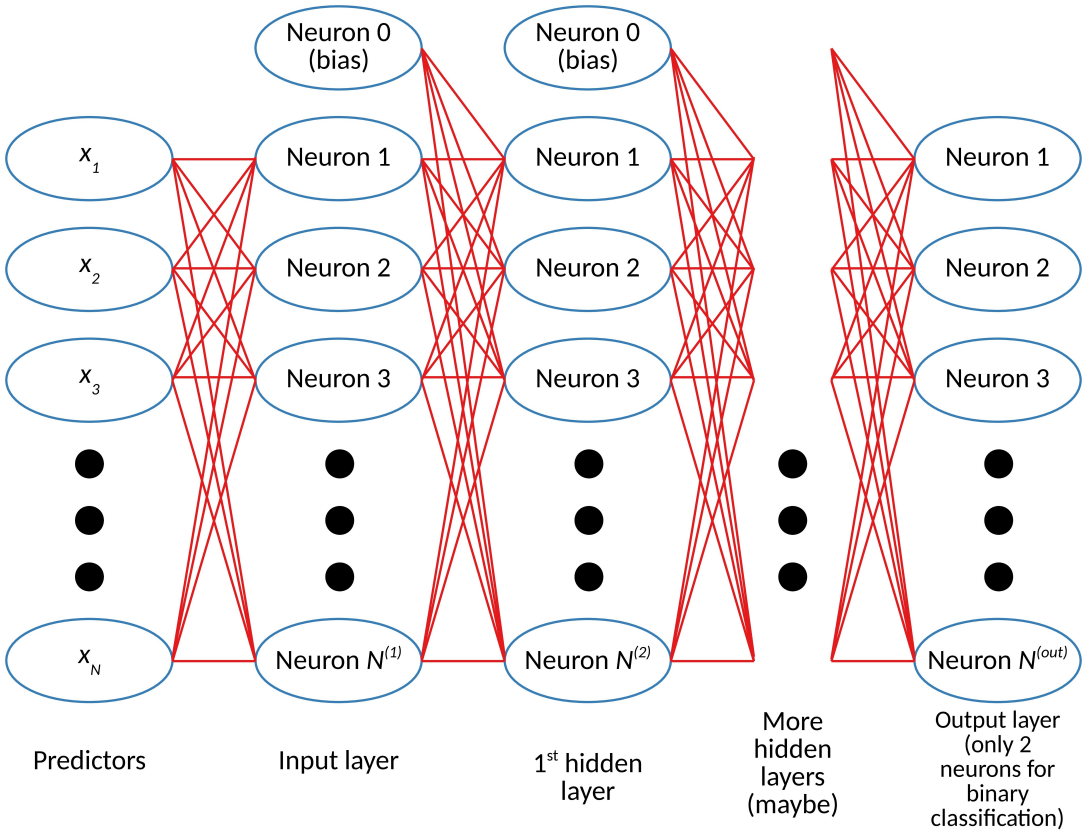

Figure 4.2: Schematic of an FFNN. In the first column, each blue ellipse is a predictor value and each red line connects a predictor value to an input neuron. In all subsequent columns, each blue ellipse is a neuron and each red line connects neurons in adjacent layers. Each red line is associated with a weight. Black (grammatical) ellipses indicate that there may be arbitrarily many predictors or neurons in each layer.

For some FFNNs (including those created by MATLAB's `patternnet`[2]), the input function for the input layer is as follows.

$$a_j{}^{(1)} = x_j \tag{4.5}$$

In other words, the input for the j^{th} input neuron is simply the value of the j^{th} predictor variable. In this case the input layer has no bias neuron, so there is no $a_0{}^{(1)}$.

There are many different activation functions. We use tansig for the hidden layers and softmax for the output layer (Figure 4.3), both of which are defaults in `patternnet`.

$$\text{tansig}(a_j{}^{(k+1)}) = \frac{2}{1 + \exp(-2a_j{}^{(k+1)})} - 1 = \frac{2}{1 + e^{(-2a_j{}^{(k+1)})}} - 1 \tag{4.6a}$$

$$\text{softmax}(a_j{}^{(k+1)}) = \frac{\exp(a_j{}^{(k+1)})}{\sum_{p=1}^{N^{(k)}} \exp(a_p{}^{(k+1)})} = \frac{e^{(a_j{}^{(k+1)})}}{\sum_{p=1}^{N^{(k)}} e^{(a_p{}^{(k+1)})}} \tag{4.6b}$$

All variables are the same as in Equation 4.4. By default, `patternnet` sets the number of neurons in the output layer to the number of classes for the predictand. Since we consider only binary classification, the output layer always has two neurons. The result of Equation 4.6b for the first neuron is the forecast probability of label = 0; the result for the second neuron is the forecast probability of label = 1. Like the output for logistic regression (Equation 4.1), the result of Equation 4.6b is bounded by $[0, 1]$, which permits this interpretation.

During training, the weights $w_{ij}{}^{(k)}$ are learned by gradient descent with backpropagation (Chapter 4 of Mitchell, 1997). For classification problems, the objective function (to be minimized by gradient descent) is usually deviance (Equation 4.2). With enough hidden layers, FFNNs usually outperform traditional methods such as LR and LREN, because they can model more complex (*i.e.,* non-linear) relationships.

[2] `http://www.mathworks.com/help/nnet/ref/patternnet.html`

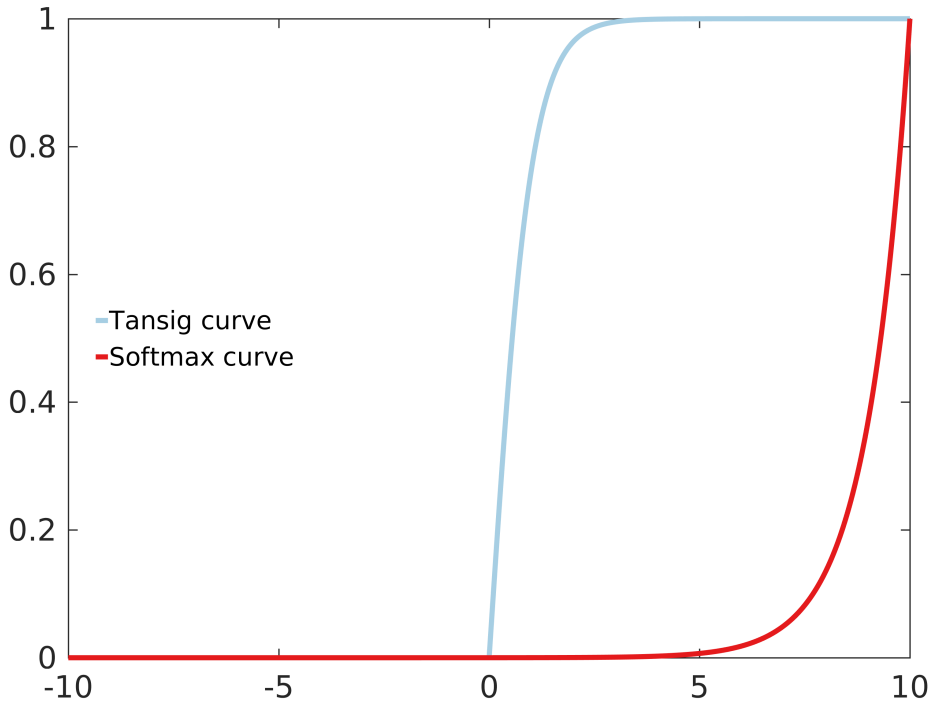

Figure 4.3: The tansig curve is plotted as $\frac{2}{1+exp(-2x)} - 1$. The softmax curve is plotted as $\frac{exp(x)}{\sum_{k=1}^{K} exp(x_k)}$, where K is the number of x-coordinates in the graph, and then scaled to $[0, 1]$.

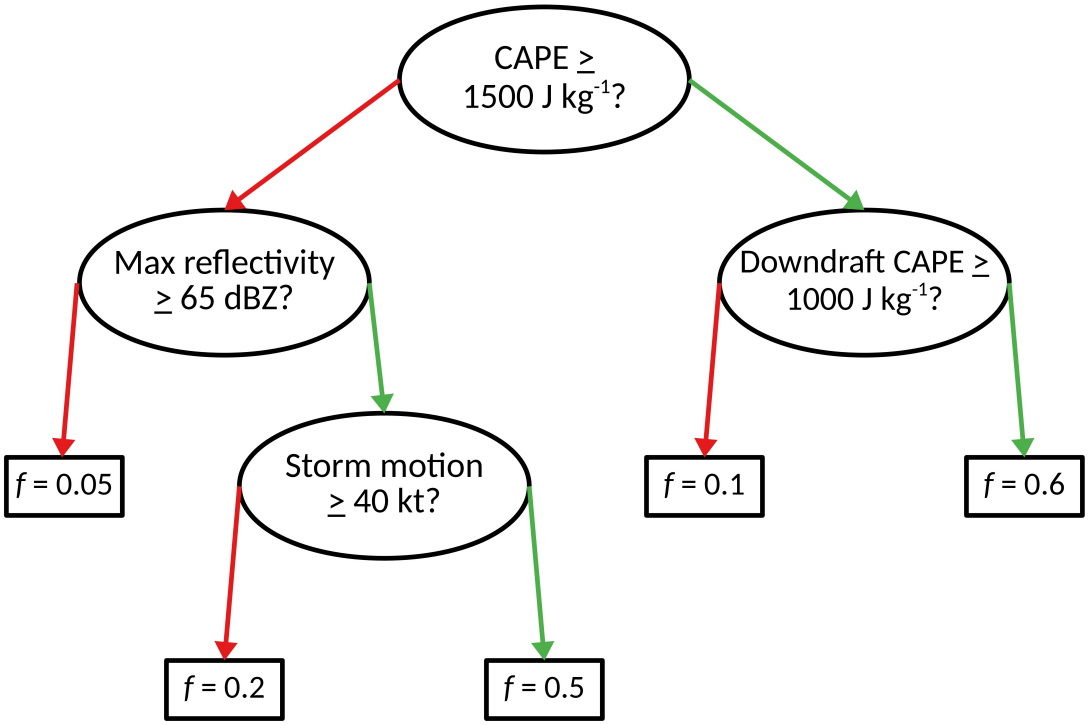

Figure 4.4: Schematic of a decision tree. At each branch node (black ellipse), a binary question is asked for one of the predictors x_j. Examples for which the answer is "yes," are sent down the right branch (green). Examples for which the answer is "no," are sent down the left branch (red). Each example eventually reaches a leaf node (black rectangle), where it is assigned a forecast f based on training examples that reached the same leaf node. This tree is only an illustrative example, not one that we actually trained.

4.1.1.4 Decision Trees

A decision tree (DT) (Chapter 3 of Mitchell, 1997) consists of several layers of nodes, where each node splits training examples into two subsets, based on a binary question. See Figure 4.4. Thus, each node is connected to one parent (in the layer above) and two children (in the layer below). The only exceptions are the root node, which has no parents, and the leaf nodes, which have no children. The DT learning procedure is given in Algorithm 4.1.

Basically, each node n receives a subset of training examples (except for the root node, which receives all training examples), each consisting of N predictor values and a label. Then the node loops over all possible values x_{jk} of each predictor variable x_j (continuous variables are discretized) and determines the best split point, which is the $\{x_j, x_{jk}\}$ pair leading to the greatest information gain (Equations 4.7). Then all examples (in the subset at node n) with $x_j < x_{jk}$ are sent to the left child, and all examples with $x_j \geq x_{jk}$ are sent to the right child. The left and right children are new nodes, which makes the function recursive. The stopping criteria ("base cases" in recursion terminology) are as follows.

1. The tree has reached some maximum depth.
2. The number of examples at node n has reached some minimum value.
3. All examples at node n have the same label.

The maximum depth and minimum number of examples are specified by the user. These parameters help to prevent overfitting. Without them, the tree would be grown until all leaf nodes are pure (*i.e.,* all leaf nodes contain examples with only one label).

For binary classification, information gain is defined as follows.

$$G = I(\frac{M_1}{M}) - \frac{M_{left}}{M} I(\frac{M_{1,left}}{M_{left}}) - \frac{M_{right}}{M} I(\frac{M_{1,right}}{M_{right}}) \tag{4.7a}$$

$$I(p) = -\left[p \log_2(p) + (1-p) \log_2(1-p)\right] \tag{4.7b}$$

G is the information gain, and I is the information function. M, M_1, M_{left}, and $M_{1,\ left}$ are – respectively – the total number of examples, number of examples with label = 1, number sent to the left child, and number sent to the left child with label = 1. From this one can easily infer what M_{right} and $M_{1,\ right}$ are. Thus, in Equation 4.7b, p is the proportion of examples with label = 1. By maximizing

information gain, the DT algorithm essentially maximizes the purity of each node (the difference between proportions of label = 0 and label = 1).

When a new example is passed through the decision tree, it will end up at a certain leaf node n^*. There are two ways to predict the label for this new example.

1. Deterministic voting. The predicted label is the mode of labels for all training examples passed to node n^*.
2. Probabilistic voting. The predicted probability of label = 1 is the proportion of training examples passed to node n^* with label = 1.

Decision trees have three major advantages. One is that they perform automated variable selection (by selecting the best split point at each node), which allows them to handle a large number of predictor variables. Second, they are easy to interpret, which is important for end users of machine-learning models. Third, they can learn complex relationships between the predictors and predictand.

Decision trees can learn complex relationships because, unlike the other algorithms described heretofore, they are not restricted to linear transformations of the training data. Although logistic regression appears non-linear, the logit function can be expressed as $\ln(\frac{f_i}{1-f_i}) = -\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij}$, so it is linear in the space formed by $\frac{f_i}{1-f_i}$ and the predictors x_j. The same is true for LREN, which is simply logistic regression with a different objective function. Also, under the input function defined in Equation 4.4, each layer of an FFNN is essentially linear (the activation functions in Equations 4.6 are linear in logarithmic space), so an FFNN is essentially a series of linear functions. Thus, decision trees are the first algorithm described heretofore that completely dispense with linear transformations. This is important, because relationships between meteorological variables are highly non-linear.

The main disadvantage of decision trees is that they are unstable, which makes them prone to overfitting. For example, suppose that A and B are storm objects with the exact same properties, except that A has 1499 J kg^{-1} of convective available potential energy (CAPE) and B has 1501 J kg^{-1} of CAPE. They both have a maximum reflectivity of 60 dBZ and downdraft CAPE of 1200 J kg^{-1}. If the decision tree in Figure 4.4 were applied to both storm objects, it would forecast $f = 0.05$ for object A and $f = 0.6$ for object B. In other words, it would forecast a 5% probability of severe wind for object A and 60% probability for object B, even though the two storm objects are essentially the same.

In general, due to the instability of decision trees, they may overfit noise in the training data, which prevents them from generalizing well to new datasets.

Algorithm 4.1: `split_node`: Used to grow decision tree.

Input: `examples` = training examples, each with N predictors and a label

`predictors` = list of N predictors; `minExamples` = min # examples at node

`currentDepth` = current depth of tree; `maxDepth` = maximum depth of tree

Output: `tree`

```
// Create new tree and check stopping criteria.
tree = new tree, consisting only of root node;
if length(examples) < minExamples ∪ all examples have same label ∪
currentDepth = maxDepth then
  return tree;
end

// Find best split point (the one with max info gain).
maxGain = -∞;
foreach predictor x_j do
  foreach possible value x_jk do
    thisGain = gain caused by splitting on x_j at x_jk;
    if thisGain > maxGain then
      maxGain = thisGain; bestPredictor = x_j; bestValue = x_jk;
    end
  end
end

leftExamples = examples where bestPredictor < bestValue;
leftSubtree = split_node(leftExamples, predictors, minExamples,
 currentDepth + 1, maxDepth);
rightExamples = examples where bestPredictor ≥ bestValue;
rightSubtree = split_node(rightExamples, predictors, minExamples,
 currentDepth + 1, maxDepth);
add leftSubtree and rightSubtree to tree; return tree;
```

4.1.1.5 Random Forests

A random forest (RF) (Breiman, 2001) is an ensemble of decision trees. Individual decision trees are trained as in Algorithm 4.1, with two exceptions.

1. Each tree is trained with a resampled set of storm objects, rather than the full training set. In general, resampling may be done with a replacement factor anywhere from $[0, 1]$. Usually this is done with a replacement factor of 1, which is equivalent to bootstrapping[3]. This procedure in general (resampling data for each tree) is called "tree-bagging".
2. At each node only a subset of the predictor variables is sampled. Thus, when looking for the best split point, the DT algorithm checks all values of $N_s \leq N$ predictor variables, rather than checking all N. This procedure is called "feature-bagging".

For a new input example, predictions from the individual trees are aggregated in one of three ways.

1. Deterministic majority vote. The predicted label is the mode of deterministic predicted labels (either 0 or 1) from the individual trees.
2. Probabilistic majority vote. The predicted probability of label = 1 is the proportion of trees with deterministic predictions of label = 1.
3. Soft voting (also probabilistic). The predicted probability of label = 1 is the mean of the individual trees' predicted probabilities of label = 1.

The goal of tree-bagging and feature-bagging is to maintain diversity among individual trees in the ensemble. If each tree is trained with different examples

[3]Henceforth, I will define "bootstrapping" as in Breiman (2001), to mean resampling with a uniform probability of replacement. This is not the same definition as in Efron (1979), who developed the bootstrap method. In Efron (1979) a statistic (*e.g.*, the mean) is "bootstrapped" by calculating it for many resampled versions of the original dataset, where resampling is done with a uniform probability of replacement.

and each split point chooses from different predictors, although individual trees will still overfit their respective training sets, they should overfit in different ways. In other words, the individual trees should have offsetting biases, which largely cancel out when they are ensembled. This allows the random forest to alleviate the overfitting problem discussed at the end of Section 4.1.1.4.

4.1.1.6 Ensembles of Gradient-boosted Trees

Gradient-boosting (GB) (Friedman, 2001) is another way to ensemble decision trees. GB starts with a constant model F_0, given by the following equation.

$$F_0(x_1, x_2, ..., x_N) = F_0(\vec{x}) = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{M} L(y_i, \gamma) \tag{4.8}$$

$F_0(\vec{x})$ is function notation, indicating that the model is a function of the predictor vector $\vec{x}$; L is the loss function; and γ is the constant that minimizes the loss function, which can be found by a line search (Section 4.8.2 of Mitchell, 1997).

When GB is used for binary classification, typical loss functions are deviance (Equation 4.2) and exponential loss. In the case of exponential loss, defined below, the algorithm is called AdaBoost (Freund and Schapire, 1997). This is short for "adaptive boosting," because the exponential loss emphasizes especially bad predictions, which makes it more adaptive to difficult training examples.

$$L_{exp} = \frac{1}{M} \sum_{i=1}^{M} \exp(-{y_i}^* f_i) = \frac{1}{M} \sum_{i=1}^{M} e^{-{y_i}^* f_i} \tag{4.9}$$

Note that y_i and ${y_i}^*$ have different meanings. $y_i = 1$ if the event (wind gust $\geq$ 50 kt) occurred and 0 otherwise, whereas ${y_i}^* = 1$ if the event occurred and -1 otherwise.

After finding γ for the constant model, GB does the following for each iteration q.

1. Compute pseudo-residuals generated by the previous model.

$$r_{iq} = -\frac{\partial L(y_i, F_{q-1}(\vec{x_i}))}{\partial F_{q-1}(\vec{x_i})} \tag{4.10}$$

r_{iq} is the pseudo-residual for the i^{th} training example, and $F_{q-1}(\vec{x_i})$ is function notation for the previous model (operating only on the predictor vector for the i^{th} training example).

2. Fit a new decision tree ("correction model") to the pseudo-residuals. Because the pseudo-residuals are real numbers and not categories, this is a regression tree, not a classification tree. Let the correction model be $h_q(\vec{x})$.
3. Add the correction model to the previous model in a weighted sum. This involves finding the best multiplier γ_q, which, like γ in Equation 4.8, can be found by a line search.

$$\gamma_q = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{M} L(y_i, F_q(\vec{x_i})) \tag{4.11a}$$

$$F_q(\vec{x_i}) = F_{q-1}(\vec{x_i}) + \gamma_q h_q(\vec{x_i}) \tag{4.11b}$$

$F_{q-1}(\vec{x_i})$, $h_q(\vec{x_i})$, and $F_q(\vec{x_i})$ are function notation for the previous model, correction model, and new model respectively.

The main difference between GB and random forests is that GB creates an additive ensemble, where the k^{th} tree is fit to the pseudo-residual from the first $k-1$ trees. Thus, trees must be trained in series. Meanwhile, a random forest is a non-additive ensemble, where all trees are fit to the label y and then ensembled afterwards. Thus, trees in a random forest can be trained in parallel.

Gradient-boosted tree (GBT) ensembles often outperform random forests, especially when AdaBoost is used, because they focus learning on difficult training examples.

4.1.2 Calibration Models

We consider two types of calibration models, which we found to be much more common in the literature than all others: Platt scaling and isotonic regression.

4.1.2.1 Platt Scaling

Platt scaling is sigmoid regression, where the single predictor is the forecast probability from the base model and the predictand is defined below.

$$y_{i,Platt} = \begin{cases} \frac{M_1+1}{M_1+2}, & \text{if } y_i = 1 \\ \frac{1}{M_0+2}, & \text{if } y_i = 0 \end{cases} \tag{4.12}$$

y_i is the label for the i^{th} training example; M_k is the number of training examples with $y_i = k$; and $y_{i,Platt}$ is the resulting label, which is similar to y_i. The use of $y_{i,Platt}$ is justified in Platt (1999).

Thus, Platt scaling fits a sigmoid curve to the base-model probabilities. The sigmoid function is also called the "logistic function," but this is not the same as the logit function (Equation 4.1) used for logistic regression. The general form of the sigmoid function is given below (see Figure 4.1).

$$f_i = \frac{1}{1 + \exp(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})} = \frac{1}{1 + e^{(-\beta_0 - \sum_{j=1}^{N} \beta_j x_{ij})}} \tag{4.13}$$

All variables are the same as in Equation 4.1. However, since there is only one predictor for Platt scaling, this can be simplified.

$$f_i = \frac{1}{1 + \exp(-\beta_0 - \beta_1 \hat{f}_i)} = \frac{1}{1 + e^{(-\beta_0 - \beta_1 \hat{f}_i)}} \tag{4.14}$$

$\hat{f}_i$ is the forecast probability from the base model, and f_i is the forecast probability from Platt scaling, for the i^{th} training example.

Since Platt scaling is a sigmoid transformation of the base-model probabilities, it works best for base models with a sigmoid-shaped reliability curve. Reliability curves (and their generalization, the attributes diagram) as a forecast-verification method will be discussed in Section 4.3.6. For the purpose of this discussion, a sigmoid-shaped reliability curve is shown in Figure 4.5. Forecasts are grouped into 10 bins, plotted on the x-axis, and for each bin the posterior

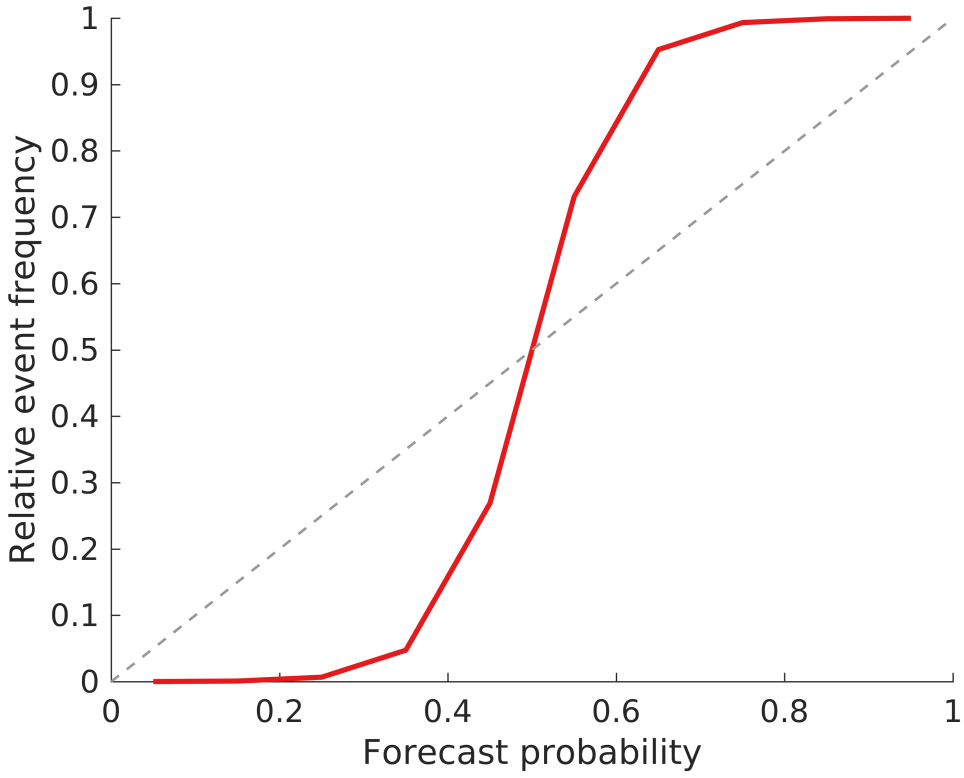

Figure 4.5: The solid red line is a sigmoid-shaped reliability curve. The dashed grey line is the perfect-reliability line. For points above this line, the predictor is "underconfident" (forecasts the event less often than it truly occurs). For points below the grey line, the predictor is "overconfident" (forecasts the event more often than it truly occurs).

probability of the event (probability of label = 1, given the forecast) is plotted on the y-axis. A sigmoid-shaped reliability curve is generated by a predictor that "hedges its bets" and does not forecast extreme probabilities, near 0 and 1.

Niculescu-Mizil and Caruana (2005) examined the reliability curves produced by 10 base models, including the six discussed in our Section 4.1.1. Of the 10 base models, only three typically produce sigmoid-shaped reliability curves: support-vector machines, GBT ensembles, and gradient-boosted stump ensembles (GBT ensembles with only two leaf nodes per tree). Thus, of the six models discussed

in Section 4.1.1, Platt scaling is appropriate only for GBT ensembles. For the other base models, a more general calibration method is required.

4.1.2.2 Isotonic Regression

Like Platt scaling, isotonic regression (IR) has only one predictor, which is the forecast probability from the base model. However, the predictand for IR is simply the label y, rather than the quantity defined in Equation 4.12. IR minimizes the Brier score (BS), subject to the constraint that $f_A \geq f_B$ for all $\hat{f}_A \geq \hat{f}_B$, where $\hat{f}_C$ is the base-model forecast for storm object C and f_C is the calibrated forecast for storm object C. In other words, isotonic regression ensures rank invariance between the base-model and calibrated forecasts.

$$\text{BS} = \frac{1}{M} \sum_{i=1}^{M} (y_i - f_i)^2 \tag{4.15}$$

M, y_i, and f_i have the same meanings as in Equation 4.2. The BS is simply the mean squared error for binary classification.

Unlike most types of regression, IR does not return an equation that can be used to predict new examples. Instead, IR returns a mapping from the base-model probabilities to calibrated probabilities. For each range of base-model probabilities, $[\hat{p}_{min}, \hat{p}_{max})$, this mapping gives the calibrated probability p. This mapping is learned by the pool-adjacent violators algorithm (PAVA) (Niculescu-Mizil and Caruana, 2005). Sample output is shown in Figure 4.6.

4.2 Sampling Techniques

We subsample data for both training and testing, for the following reasons.

1. The 804 training/testing days (Section 3.2) have a total of 19,965,275 storm objects. Predictors and labels for these storm objects take up 60.4 GB of memory, and intermediate data created during training would take up even

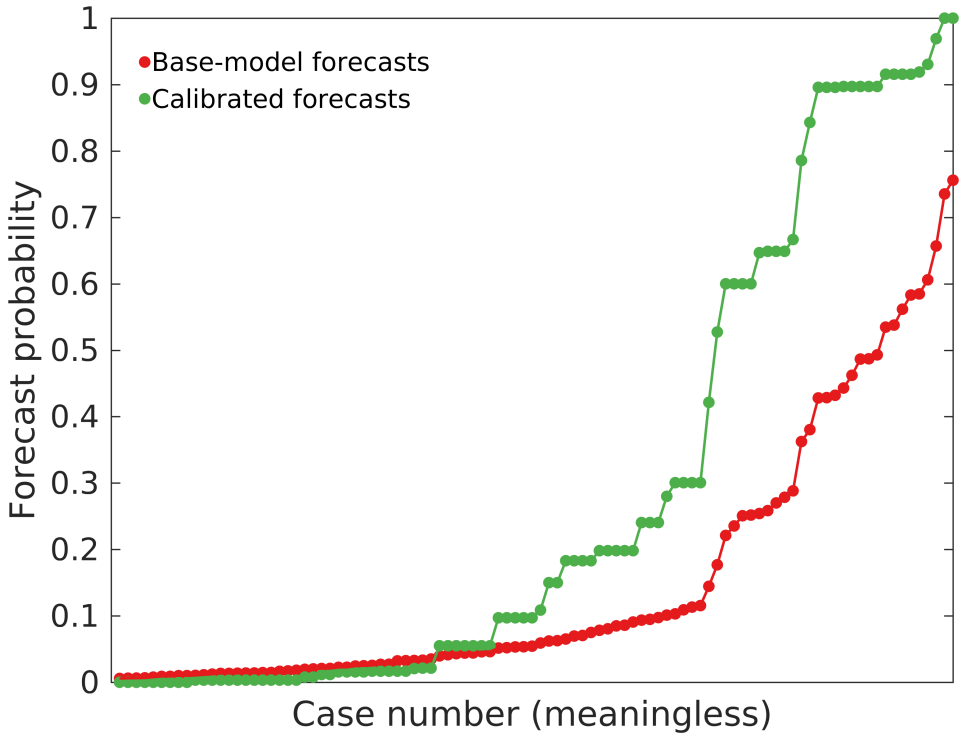

Figure 4.6: Results of isotonic regression for 100 testing examples (independent of training data, using the 24-hour criterion discussed in Section 4.2.4). The x-axis is used only to order the forecasts, so forecasts at the same x-position are for the same storm object. Before plotting, the base-model forecasts were sorted from left to right. Calibrated forecasts also increase from left to right, because isotonic regression ensures rank invariance between the base-model and calibrated forecasts.

more memory. The computers used for this project have no more than 64 GB of RAM, so training with all data would have required memory-shuffling (as for `w2besttrack` in Section 3.3). However, the runtime would have still been extremely long (days to train a single model), and this would have been exacerbated by the input-output operations associated with memory-shuffling.

2. The distribution of maximum storm-object winds (U_{max}) is highly skewed. For a buffer distance of 0 km and lead time of $[60, 90]$ minutes, the proportion of $U_{max} \geq 50$ kt (thus, the proportion of label = 1) is only 0.03% (Figure 4.7a). For a buffer distance of 10 km and lead time of $[0, 15]$ minutes, the proportion of $U_{max} \geq 50$ kt is 0.21% (Figure 4.7b). These are the minimum and maximum proportions, respectively, over all combinations of buffer distance and lead time.

   In general, machine-learning algorithms do not learn well from highly skewed data. This is because the objective function is dominated by the large number of negative examples (with label = 0) and relatively insensitive to the small number of positive examples (with label = 1). Thus, the objective function does not suffer much from simply predicting zero (or very small probabilities) for all examples. Although the model could increase its sharpness by predicting higher probabilities for some examples, this would likely improve predictions for a small number of positive examples and worsen predictions for a very large number of negative examples, leading to a worse value of the objective function.

3. The distribution of U_{max} in the full dataset is certainly not representative of the real-world distribution. U_{max} is based only on wind observations from weather stations and humans, which are often insufficient to resolve the maximum winds produced by a storm cell. For example, if storm object

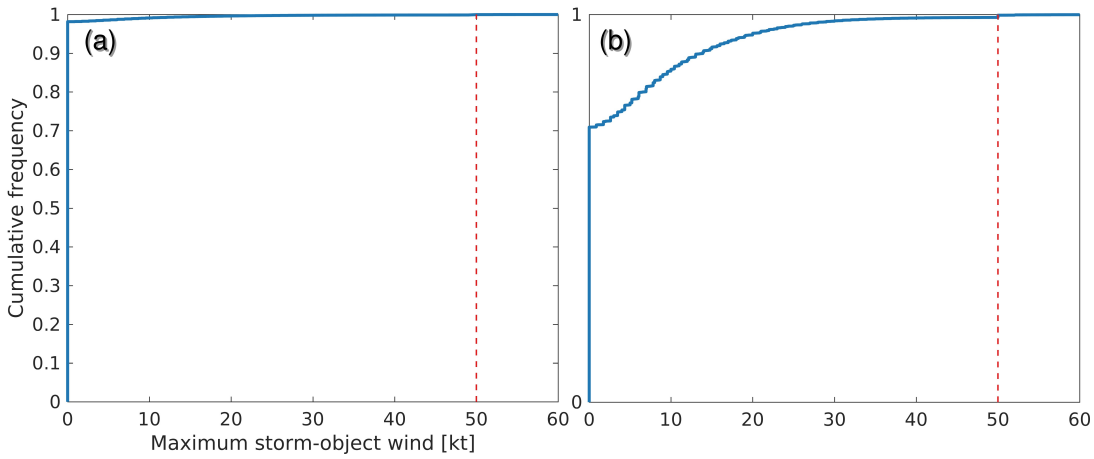

Figure 4.7: CDF of U_{max} in full dataset (no subsampling) for (a) buffer distance of 0 km and lead time of $[60, 90]$ minutes; (b) buffer distance of 10 km and lead time of $[0, 15]$ minutes. If a storm object has no wind observations over the given buffer distance and lead-time window, its maximum wind is assigned as 0.

> S is linked to only one wind observation (of < 50 kt), one cannot say with certainty that S never produced severe winds. It is quite likely that severe winds occurred and simply were not observed.

In the following subsections, sampling is done independently for each buffer distance and lead-time window.

4.2.1 Training the Base Model

To train the base model, we sample uniformly from the distribution of U_{max}. Specifically, for each buffer distance d and lead-time window $[\Delta t_{min}, \Delta t_{max}]$, we draw an equal number of storm objects from each of the following categories. See Figure 4.8.

- $U_{max} < 10$kt
- $U_{max} \in [10, 20)$ kt
- $U_{max} \in [20, 30)$ kt

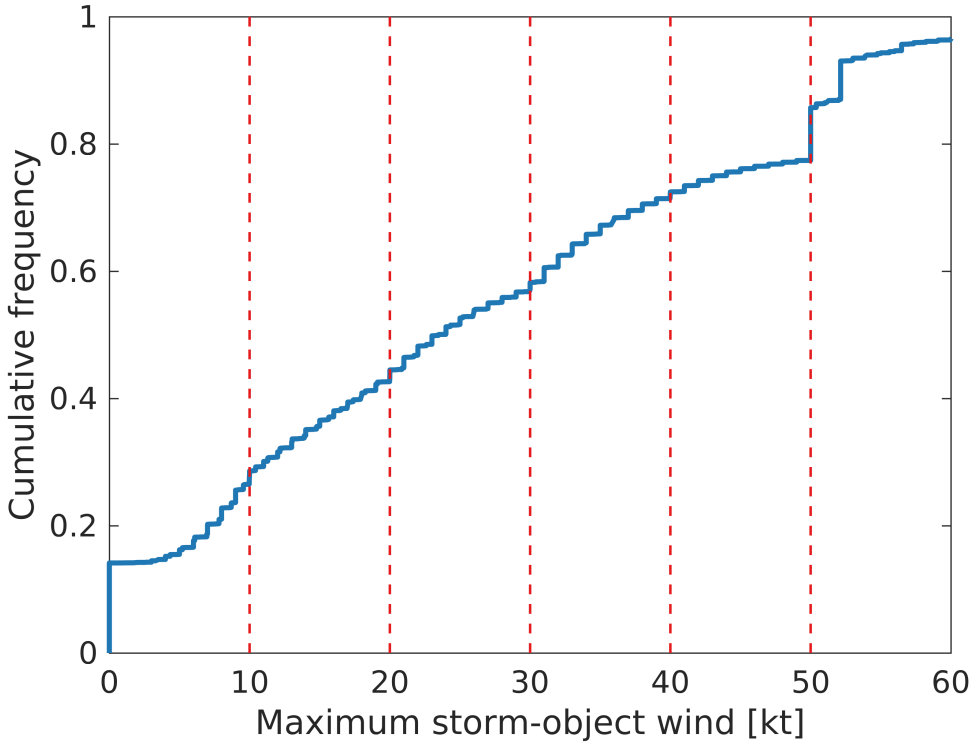

Figure 4.8: CDF of U_{max} after uniform sampling, for buffer distance of 5 km and lead time of $[30, 45]$ minutes (median values).

- $U_{max} \in [30, 40)$ kt
- $U_{max} \in [40, 50)$ kt
- $U_{max} \geq 50$kt
- U_{max} undefined (because the storm no longer exists after Δt_{min}; in other words, the storm is dead)

The number of storm objects drawn from each category is the minimum over all categories. In other words,

$$M_{min} = \min_k M_k, \tag{4.16}$$

where M_k is the number in the k^{th} category. The M_{min} most observed storm objects (*i.e.,* those with the greatest numbers of linked wind observations) are drawn from each category. Clearly the most-observed criterion is irrelevant for the least common category, which has only M_{min} storm objects, and the dead-storm category, which has no wind observations linked with storm objects after Δt_{min} (the minimum lead time). However, for all other categories, this ensures that the most observed storm objects (*i.e.,* those with the least uncertainty in U_{max}) are used.

Sampling uniformly with respect to U_{max} allows the model to learn from storms with all values of U_{max}, which should allow it to predict new storms with all values of U_{max}. Allowing for dead storms is important, because it eliminates "survivor bias" in the model. If a model is trained only with storms that still exist after Δt_{min}, it will learn to predict U_{max} only for these storms. In other words, the model will learn the conditional probability $p(U_{max} \geq 50 \text{ kt} \mid \text{storm exists after } \Delta t_{min})$, rather than the simple probability $p(U_{max} \geq 50\,\text{kt})$. Our goal is to predict the simple probability, because predicting storm longevity itself is non-trivial.

4.2.2 Training the Calibration Model

As mentioned in Section 4.1, most base models generate poorly calibrated probabilities. Also, even for those that do not, we have ensured poor calibration by training with an unrealistic distribution of U_{max}. The calibration model should be trained with a probability distribution similar to the real world. As mentioned in Section 4.2, the full dataset is not representative of the real world, so random sampling from the full dataset would not suffice here. Instead, we use the best-observed storms, for which we can be confident that the label is accurate. "Best-observed" is defined by the following criteria.

1. If $U_{max} \geq 50$ kt, we can be certain that the true label is 1, so the storm object is included.
2. If the number of linked wind observations reaches some minimum value N^*_{obs}, we can be confident in the assigned label either way (whether it is 0 or 1), so the storm object is included.
3. The proportion of dead storms (those that do not exist after Δt_{min}) in the training set should equal that in the full dataset. Thus, after steps 1 and 2, enough dead storms are randomly selected to make the proportions equal.

$$
\begin{aligned}
p^*_{dead} &= p_{dead} \\
\Rightarrow \frac{M^*_{dead}}{M^*} &= p_{dead} \\
\Rightarrow \frac{M^*_{dead}}{M^*_{dead} + M^*_{alive}} &= p_{dead} \\
\Rightarrow M^*_{dead} &= p_{dead}(M^*_{dead} + M^*_{alive}) \\
&= p_{dead}M^*_{dead} + p_{dead}M^*_{alive} \\
\Rightarrow (1 - p_{dead})M^*_{dead} &= p_{dead}M^*_{alive} \\
\therefore M^*_{dead} &= \frac{p_{dead}}{1 - p_{dead}} M^*_{alive}
\end{aligned}
\tag{4.17}
$$

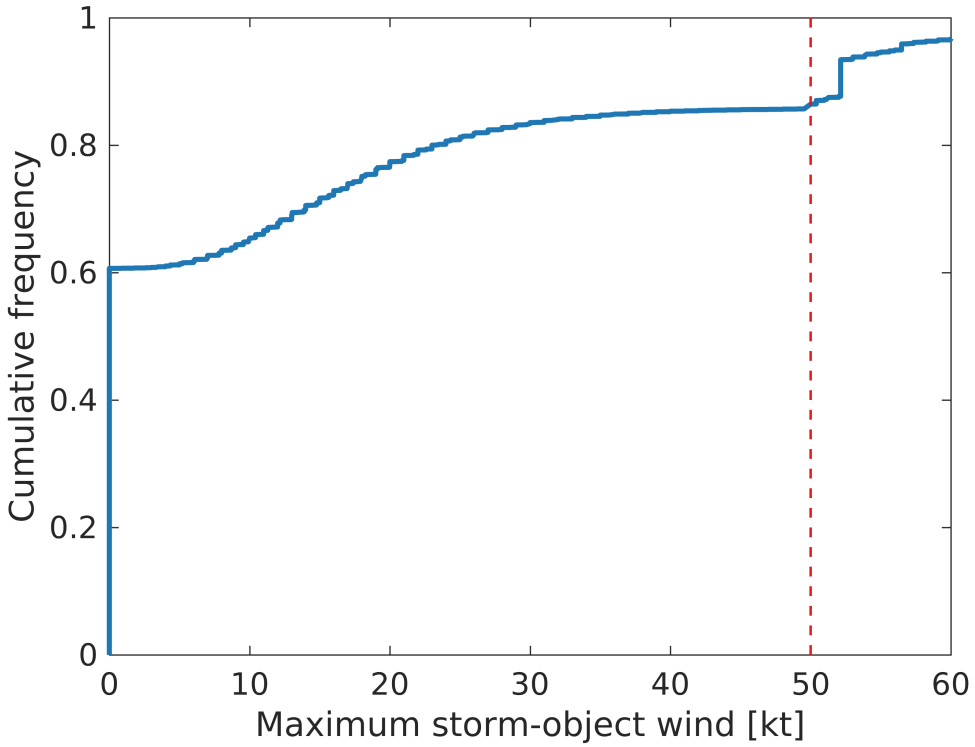

Figure 4.9: CDF of U_{max} after sampling from best-observed distribution, for buffer distance of 5 km and lead time of $[30, 45]$ minutes. Again, if a storm object has no wind observations over the given buffer distance and lead-time window (possibly because the storm is dead), its maximum wind is assigned as 0.

p_{dead} is the proportion of dead storms in the full dataset; p^*_{dead} is the proportion of dead storms in the sampled dataset; M^*_{dead} is the raw number of sampled storm objects that are dead after Δt_{min}; and M^*_{alive} is the raw number of sampled storm objects that are still alive after Δt_{min}. Thus, M^*_{dead} is the number of dead storms needed in the sampled dataset to make $p^*_{dead} = p_{dead}$. This step is necessary to eliminate survivor bias from the calibration model. See Figure 4.9 for a typical CDF from this sampling method.

4.2.3 Testing the Combined Model

The "combined model" is the base model after calibration. As in training the calibration model, the U_{max} distribution used to test the combined model should be similar to the real world, so that testing results indicate how the model would perform in a real-world setting (*e.g.,* in a forecasting office). Thus, we use the best-observed distribution (Section 4.2.2) for testing as well.

4.2.4 Independence of Sample Sets

There are three sample sets for each combined model: the base-model-training set, calibration-model-training set, and testing set. To ensure that the combined model can be generalized to new data, the three sets should be mutually independent. To ensure independence, an example used in one set cannot occur within 24 hours of an example used in another set. This prevents two types of contamination among the sample sets.

1. Storm-track contamination. This occurs when storm objects in the same track are split among multiple sets. If one storm object in track S^* is linked with severe winds, it is likely that other objects in S^* are linked with severe winds.
2. Storm-environment contamination. This occurs when storm cells from the same environment are split among multiple sets. If one storm cell on day n produced severe winds, it is likely that other cells on day n produced severe winds, since they occurred in similar environments. Of course this is not true for storm cells separated by long distances (*e.g.,* one in Tuktoyaktuk and another in Qikiqtarjuaq) or time periods (*e.g.,* one at 0600 UTC and another at 1800 UTC). Thus, we could have developed a sharper (less cautious) independence criterion. However, we decided against this, since

the 24-hour criterion is clearly sufficient and does not lead to many storm objects being unused.

4.3 Forecast Verification

Forecast verification (FV) is a rich field of study: hundreds of methods have been developed for binary classification alone. Thus, in deciding which FV methods to use, we approach the problem in a systematic way. All FV methods are based on some property of the joint distribution of forecasts and observations, so we start with this joint distribution.

4.3.1 Joint Probability-density Functions

The most fundamental way to verify forecasts is to plot all forecast-observation pairs. However, this is rarely done in practice, because most forecast models are verified with a large number of examples, which makes such analysis both time-consuming and difficult to generalize into conclusions. For any FV method, there is a trade-off between the information content and "glance value" (how much a human analyst can conclude from output produced by said method). Plotting all forecast-observation pairs has high information content but low glance value.

Thus, in practice the most fundamental way to verify forecasts is to plot the joint probability-density function (PDF) of the forecasts and observations (Figure 4.10). The joint PDF estimates the proportion of examples occurring in each region of the forecast-observation space.

$$p(f_{ij}, y_{km}) = p(f \in [f_i, f_j] \cap y \in [y_k, y_m]) \tag{4.18}$$

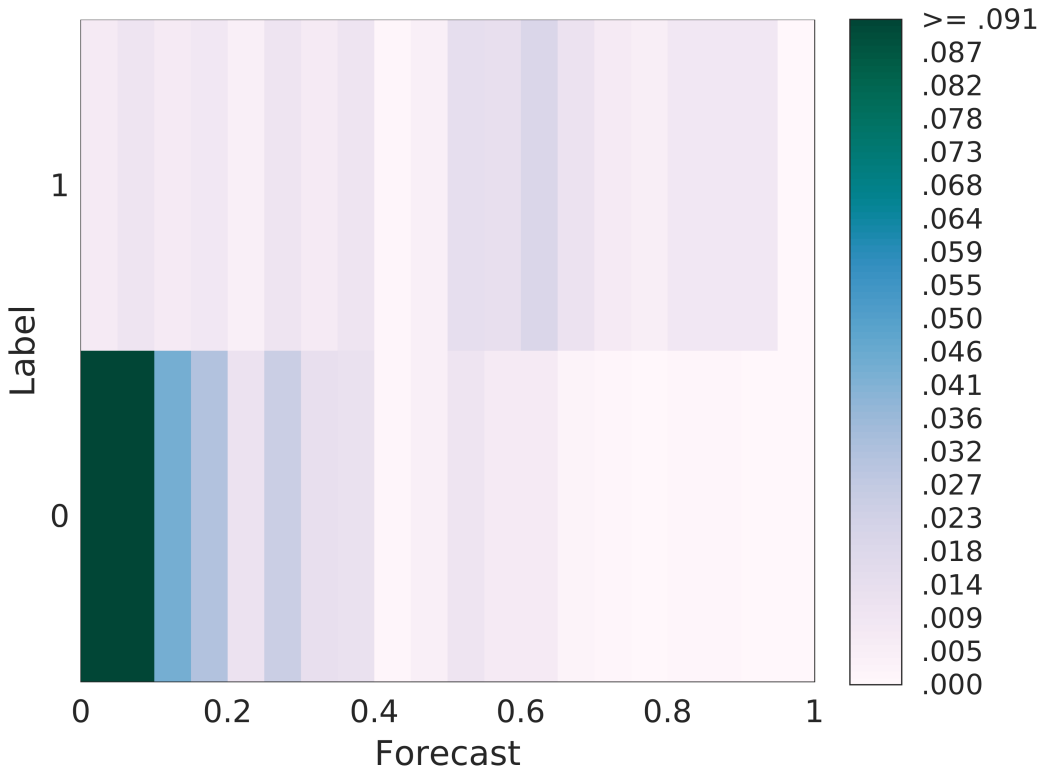

Figure 4.10: Basic joint PDF. The colour in region $f \in [f_i, f_j] \cap y = y_k$ indicates the proportion of examples falling in this region.

f is the forecast; $[f_i, f_j]$ is the region of interest for the forecast; y is the observation; and $[y_k, y_m]$ is the region of interest for the observation. For binary classification there are only two possible values of y, so this can be simplified.

$$p(f_{ij}, y_k) = p(f \in [f_i, f_j] \cap y = y_k) \tag{4.19}$$

If the testing data are independent of the training data (discussed in Section 4.2.4) and distributed similarly to real-world data (discussed in Section 4.2.3), $p(f_{ij}, y_k)$ is also the proportion of real-world examples expected in region $f \in [f_i, f_j] \cap y = y_k$. In the following subsections, when referring to the joint PDF, notation will be simplified to $p(f, y)$. Keep in mind that, in order to evaluate the joint PDF, values of f_i, f_j, and y_k are needed.

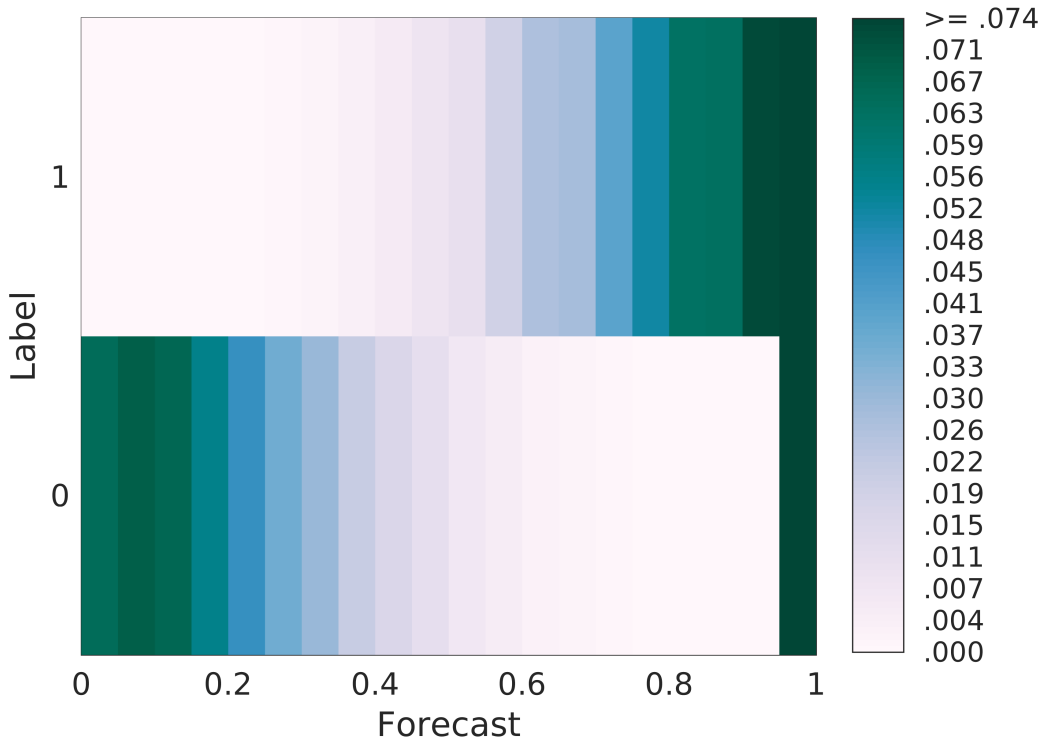

Figure 4.11: Similar to Figure 4.10, except that there is an obvious problem in the bottom-right of the distribution.

The advantage of the joint PDF is that it highlights regions of the forecast-observation space where examples occur most and least frequently. This can help to diagnose specific problems with the model. For instance, if there are many examples in the region $f \in [0.95, 1.00] \cap y = 0$ (as in Figure 4.11), this means that many high-probability forecasts are made for non-events. Perhaps this is because there is a bug in the code, inverting either the forecasts or the observations. Such a hypothesis would be difficult to draw from an FV method with less information content.

4.3.2 Aspects of Forecast Quality

Murphy (1993) discusses 10 aspects of forecast quality. These are defined in Table 4.1 (adopted from Murphy's Table 2). Table 4.1 highlights two factorizations of the joint PDF, which will be used in the following subsections: the calibration-refinement (CR) factorization and the likelihood–base-rate (LBR) factorization. These factorizations are shown in Equations 4.20a and 4.20b, respectively.

$$p(f, y) = p(f)p(y \mid f) \tag{4.20a}$$

$$p(y, f) = p(y)p(f \mid y) \tag{4.20b}$$

From Table 4.1, resolution and reliability are clearly related to the CR factorization; discrimination I and II are clearly related to the LBR factorization. The other six aspects of forecast quality do not depend on these factorizations.

Aspect	**Definition**	**Distributions**	**As Shown in**
Accuracy	Mean correspondence between forecasts and observations. For deterministic forecasts, frequency of agreement between forecasts and observations.	$p(f, y)$	ROC curve
Association	Strength of linear relationship between forecasts and observations.	$p(f, y)$	Joint PDF

Continued on next page

Table 4.1 – continued from previous page

Aspect	Definition	Distributions	As Shown in
Bias	Correspondence between mean forecast and mean observation.	$p(f)$ and $p(y)$	Performance diagram
Discrimination I	Correspondence between conditional mean forecast and conditioning observation, averaged over all observations.	$p(y)$ and $p(f \mid y)$	ROC curve
Discrimination II	Correspondence between conditional mean forecast and unconditional mean forecast, averaged over all observations.	$p(y)$ and $p(f \mid y)$	ROC curve
Reliability	Correspondence between conditional mean observation and conditioning forecast, averaged over all forecasts.	$p(f)$ and $p(y \mid f)$	Attributes diagram
Resolution	Difference between conditional mean observation and unconditional mean observation, averaged over all forecasts.	$p(f)$ and $p(y \mid f)$	Attributes diagram

Continued on next page

Table 4.1 – continued from previous page

Aspect	Definition	Distributions	As Shown in
Sharpness	Variability of forecasts.	$p(f)$	Attributes diagram (forecast histogram)
Skill	Value of some verification statistic, relative to a baseline (*e.g.,* climatology or another forecast model).	$p(f, y)$	Attributes diagram (no-skill line and positive-skill area)
Uncertainty	Variability of observations.	$p(y)$	Attributes diagram

Table 4.1: Aspects of forecast quality. This table is adapted from Murphy's (1993) Table 2. Mathematical notation and vocabulary have been adjusted to suit this document.

Event Forecast	Event Observed	
	yes	no
yes	*a*	*b*
no	*c*	*d*

Figure 4.12: Basic contingency table. Entries a, b, c, and d are explained in the text.

4.3.3 Contingency Tables

Almost all verification statistics for binary classification are based on the contingency table (CT), or "confusion matrix". This is a two-by-two matrix with the following entries (Figure 4.12).

- Top-left corner (1, 1). a = number of true positives (where the event was both forecast and observed).
- Top-right corner (1, 2). b = number of false positives (where the event was forecast but not observed).
- Bottom-left corner (2, 1). c = number of false negatives (where the event was observed but not forecast).
- Bottom-right corner (2, 2). d = number of correct negatives, or "correct nulls" (where the event was neither forecast nor observed).

This table is clearly missing one type of information, which is the probability associated with each forecast. To transform the probabilistic forecasts into deterministic ones, a threshold f^* is defined so that

$$\hat{y} = \begin{cases} 1, & \text{if } f \geq f^* \\ 0, & \text{if } f < f^* \end{cases} \tag{4.21}$$

where f is the forecast probability and $\hat{y}$ is the resulting deterministic forecast.

There are eight "default" (single-row or single-column) statistics based on the CT, defined below.

$$\text{POD} = \frac{a}{a+c} \tag{4.22a}$$

$$\text{FOM} = \frac{c}{a+c} = 1 - \frac{a}{a+c} = 1 - \text{POD} \tag{4.22b}$$

$$\text{POFD} = \frac{b}{b+d} \tag{4.22c}$$

$$\text{NPV} = \frac{d}{b+d} = 1 - \frac{b}{b+d} = 1 - \text{POFD} \tag{4.22d}$$

$$\text{SR} = \frac{a}{a+b} \tag{4.22e}$$

$$\text{FAR} = \frac{b}{a+b} = 1 - \frac{a}{a+b} = 1 - \text{SR} \tag{4.22f}$$

$$\text{DFR} = \frac{c}{c+d} \tag{4.22g}$$

$$\text{FOCN} = \frac{d}{c+d} = 1 - \frac{c}{c+d} = 1 - \text{FOCN} \tag{4.22h}$$

POD is the probability of detection, or "hit rate"; FOM is the frequency of misses; POFD is the probability of false detection; NPV is the negative predictive value; SR is the success ratio; FAR is the false-alarm ratio; DFR is the detection-failure ratio; and FOCN is the frequency of correct nulls. Column-based statistics (4.22a-4.22d) are related to the likelihood–base-rate (LBR) factorization, and row-based statistics (4.22e-4.22h) are related to the calibration-refinement (CR) factorization (Section 4.3.2). All of these statistics range from $[0, 1]$. For the FOM, POFD, FAR, and DFR, lower values are considered better. For the POD, NPV, SR, and FOCN, higher values are considered better.

There are two more CT-based statistics used in this study.

$$\text{FB} = \frac{a+b}{a+c} \tag{4.23a}$$

$$\text{CSI} = \frac{a}{a+b+c} \tag{4.23b}$$

FB (frequency bias) is the number of forecast events over the number of actual events, while CSI (critical-success index) is the accuracy ($\frac{a+d}{a+b+c+d}$) with correct

nulls removed. This is useful, because correct nulls are often trivial to predict and difficult to count. For example, it is difficult to prove, based on limited surface observations, that a storm cell never produced severe winds. FB ranges from $[0, \infty)$, with values near 1 considered better. CSI ranges from $[0, 1]$, with higher values considered better.

The main disadvantage of the CT and CT-based statistics is that they depend on deterministic forecasts, which are less informative than probabilistic forecasts. However, this disadvantage can be alleviated by creating a CT for each probability threshold, which is the basis of Sections 4.3.4 and 4.3.5.

4.3.4 Receiver-operating-characteristic Curves

For several probability thresholds (used in Equation 4.21 to convert probabilistic to deterministic forecasts), the receiver-operating-characteristic (ROC) curve plots probability of detection (Equation 4.22a) on the y-axis and probability of false detection (Equation 4.22c) on the x-axis. See Figure 4.13. In other words, the ROC curve plots POD as a function of POFD, both of which decrease from 1 to 0 as the threshold is increased from 0 to 1.

To describe the quality of a ROC curve, the area under the ROC curve (AUC) is used as a verification statistic. The following criteria are typically associated

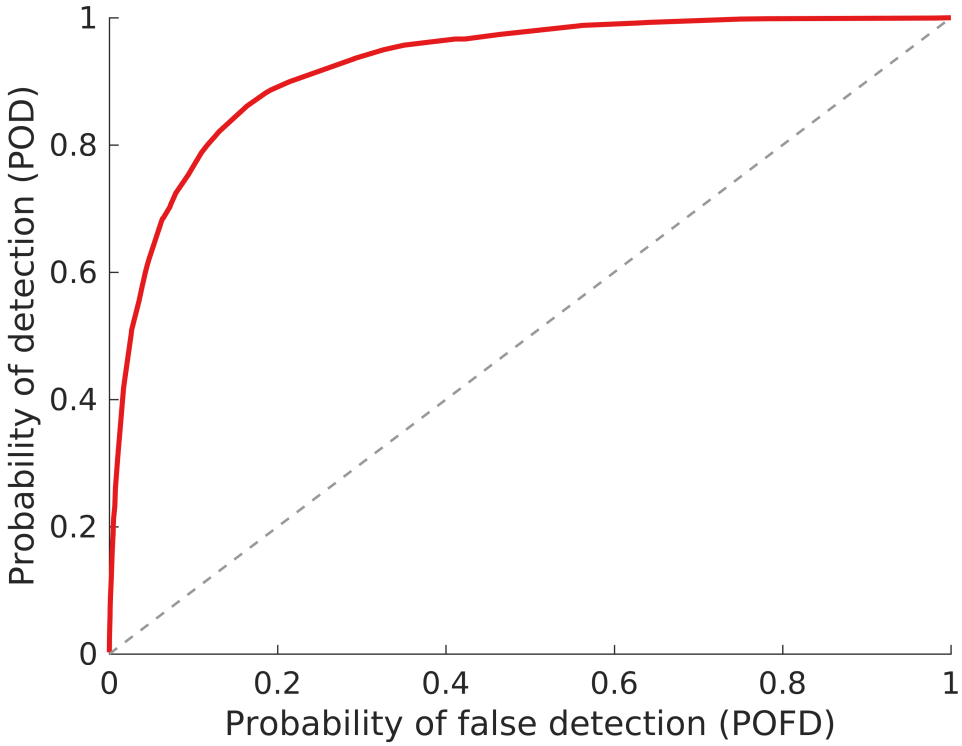

Figure 4.13: The solid red line is the ROC curve. The dashed grey line is the perfect-reliability line. For points above this line, the predictor has positive skill (better than a random predictor). For points below the grey line, the predictor has negative skill (worse than a random predictor).

with different AUC values (*e.g.,* Luna-Herrera et al., 2003; Muller et al., 2005; Mehdi et al., 2011).

$$\text{AUC} \begin{cases} = 1.0, & \text{perfect} \\ \in [0.9, 1.0), & \text{excellent} \\ \in [0.8, 0.9), & \text{good} \\ \in [0.7, 0.8), & \text{fair} \\ \in [0.6, 0.7), & \text{poor} \\ < 0.6, & \text{unacceptable} \end{cases}$$

The ROC curve has two main advantages. One is that it plots values for many probability thresholds, which allows an optimal threshold to be found for Equation 4.21. Also, it is insensitive to the class distribution (*i.e.,* the relative proportions of events and non-events – in our case, the relative proportions of $U_{max} \geq 50$ kt and $U_{max} < 50$ kt), so large AUC values can be achieved even for highly skewed datasets. However, this is also a disadvantage, because it obfuscates issues that most forecast models encounter with highly skewed datasets. In particular, for extremely rare events, the success ratio (Equation 4.22e) tends to be very low and false-alarm ratio (Equation 4.22f) tends to be very high. For extremely common events, the frequency of correct nulls (Equation 4.22h) tends to be very low and detection-failure ratio (Equation 4.22g) tends to be very high. Since these statistics are based on the CR factorization, they are not shown in ROC curves, which are based on the LBR factorization (see Section 4.3.2 for discussion of factorizations).

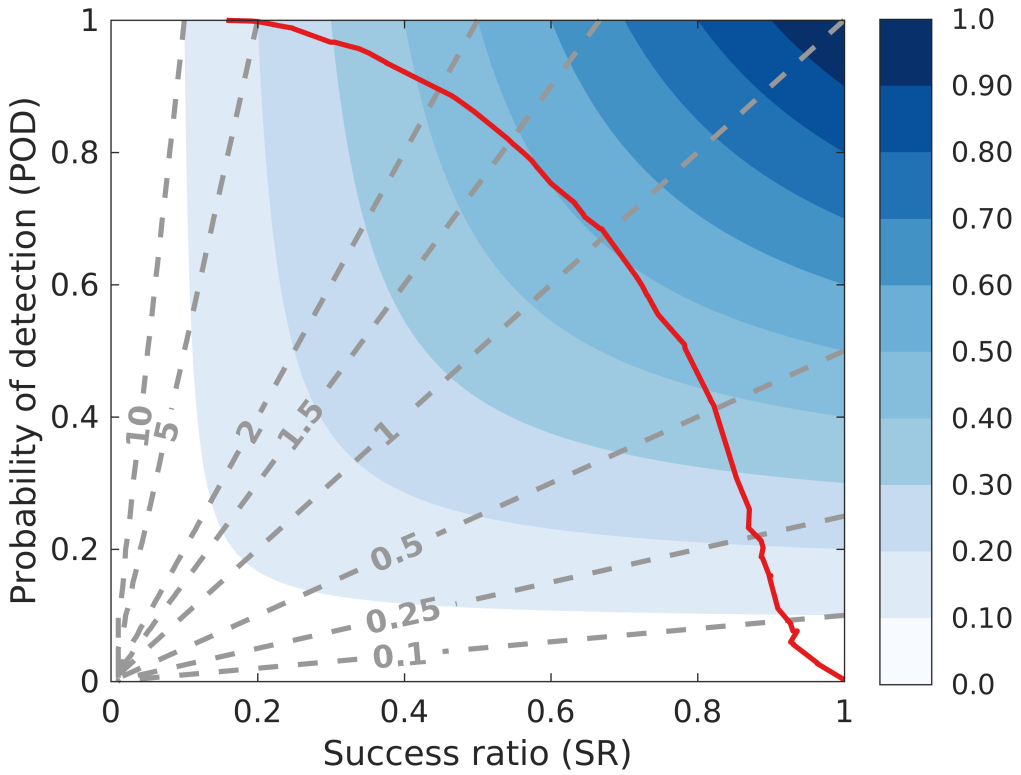

Figure 4.14: Basic performance diagram. The solid red line is the performance curve. The dashed grey lines are FB contours. The colour fill is CSI.

4.3.5 Performance Diagrams

The performance diagram is an extension of the performance curve. For several probability thresholds (f^* in Equation 4.21), the performance curve plots probability of detection (Equation 4.22a) on the y-axis and success ratio (Equation 4.22e) on the x-axis. See Figure 4.14. In other words, the performance curve plots POD as a function of SR. As the threshold increases from 0 to 1, the POD decreases from 1 to 0, while the SR increases from 0 to 1. This is unlike the ROC curve, where both POD and POFD decrease. This is why the performance curve looks like a ROC curve that has been flipped left-to-right.

In addition to this curve, the performance diagram contains a set of CSI and FB contours. Equations 4.24 and 4.25 are used to transform FB and CSI into POD-SR space.

$$\begin{aligned}
\mathrm{FB} &= \frac{a+b}{a+c} \\
&= \frac{a+b}{a+c}\frac{a}{a} \\
&= \frac{a+b}{a}\frac{a}{a+c} \\
&= (\frac{a}{a+b})^{-1}\frac{a}{a+c} \\
&= (\mathrm{SR})^{-1}\mathrm{POD} \\
\therefore \mathrm{FB} &= \frac{\mathrm{POD}}{\mathrm{SR}}
\end{aligned} \tag{4.24}$$

$$\begin{aligned}
\mathrm{CSI} &= \frac{a}{a+b+c} \\
\Rightarrow (\mathrm{CSI})^{-1} &= \frac{a+b+c}{a} \\
&= \frac{a+b}{a} + \frac{c}{a} \\
&= \frac{a+b}{a} + (\frac{a+c}{a} - \frac{a}{a}) \\
&= \frac{a+b}{a} + \frac{a+c}{a} - 1 \\
&= (\frac{a}{a+b})^{-1} + (\frac{a}{a+c})^{-1} - 1 \\
&= (\mathrm{SR})^{-1} + (\mathrm{POD})^{-1} - 1 \\
\therefore \mathrm{CSI} &= \frac{1}{\frac{1}{\mathrm{SR}} + \frac{1}{\mathrm{POD}} - 1}
\end{aligned} \tag{4.25}$$

To our knowledge, the statistics most commonly used to describe the quality of a performance diagram are the maximum CSI and FB at maximum CSI.

$$\mathrm{CSI}_{\max} = \max_{f^*} \mathrm{CSI}(f^*) \tag{4.26a}$$

$$\begin{cases}
\mathrm{FB}_{\mathrm{best}} = \mathrm{FB}(f^*_{best}) \\
f^*_{best} = \underset{f^*}{\operatorname{argmax}}\, \mathrm{CSI}(f^*)
\end{cases} \tag{4.26b}$$

f^* is a probability threshold from Equation 4.21; f^*_{best} is the probability threshold at maximum CSI; and FB_{best} is the FB at maximum CSI. CSI(f^*) and FB(f^*_{best}) are function notation, to indicate that the two quantities are functions of f^*.

Unlike for AUC, there is no standard set of evaluative criteria for CSI. This is because values are highly dependent on the class distribution, which varies from problem to problem. For some problems (*e.g.*, Rico-Ramirez and Cluckie, 2008) a CSI of 0.8 is considered poor, and for some (*e.g.*, Gagne et al., 2015) a CSI of 0.4 is considered good. However, in all cases the CSI-maximizing frequency bias should be close to 1 (*e.g.,* between 0.75 and 1.25). Probablity calibration (Section 4.1.2), which is verified with the attributes diagram (Section 4.3.6), can help to move this FB closer to 1.

The main advantage of the performance diagram is that it shows much of the information missing from the ROC curve. This is because the performance diagram is based on both the CR and LBR factorizations (see Section 4.3.2), whereas the ROC curve is based only on the LBR factorization. The main disadvantage of the performance diagram is that it does not explicitly show probabilistic information, so it cannot be used to verify the reliability (see Table 4.1) of the model.

4.3.6 Attributes Diagrams

The attributes diagram (Figure 4.15) is an extension of the reliability curve, which is used to verify probabilistic forecasts. The reliability curve divides the forecast probabilities into 10 bins and, for each bin, plots the mean forecast probability (bin center) on the x-axis and conditional event frequency on the y-axis. In other words, the reliability curve plots $p(y = 1 \mid f)$ as a function of f. Thus, the reliability curve uses the CR factorization.

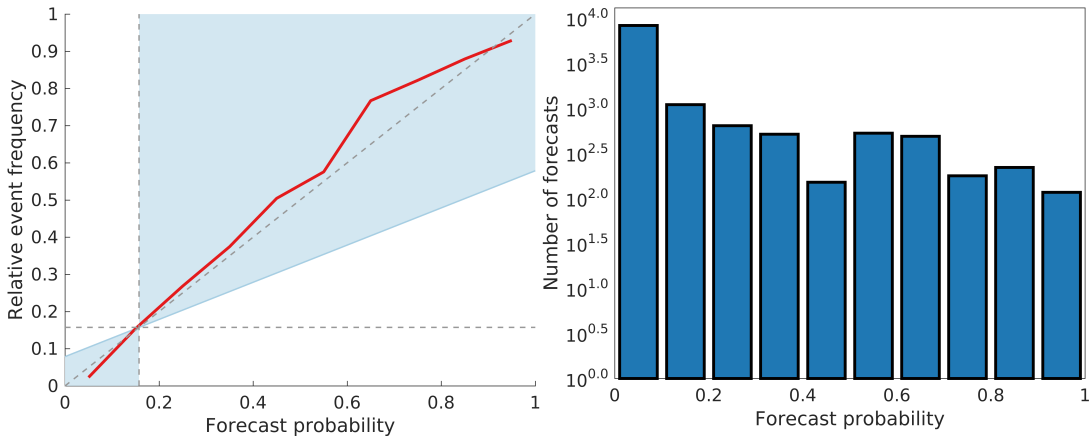

Figure 4.15: Basic attributes diagram. The solid red line is the reliability curve. All other lines are explained in the text.

In addition to the reliability curve, the attributes diagram contains a forecast histogram and a set of reference lines. The forecast histogram shows the PDF of forecasts only, *p(f)*, regardless of the observations. Thus, the histogram can be used to verify the sharpness of the model (see Table 4.1). The histogram also indicates how many examples were used to create each point in the reliability curve, which can be used to estimate uncertainty. However, this can also be done by bootstrapping, which we do for all experimental results (Section 6.1).

The four reference lines in the attributes diagram are defined below.

1. Perfect-reliability line. This is the $x = y$ line, where the conditional event frequency always equals the forecast probability, or $p(y = 1 \mid f) = f$.
2. Climatology line. This is the vertical line at $x = \bar{y}$, where $\bar{y}$ is the unconditional (climatological) event frequency.
3. No-resolution line. This is the horizontal line at $y = \bar{y}$, or the horizontal version of the climatology line. For a model with no resolution (*i.e.*, where the conditional event frequency is $\bar{y}$ for all forecasts), the reliability curve will coincide with this line.

4. No-skill line. This is the diagonal line in light blue, passing from ($x = 0, y = \frac{1}{2}\bar{y}$) to ($x = 1, y = \frac{1}{2}[1 + \bar{y}]$). Where the reliability curve passes through the shaded area, the model has a positive Brier skill score (BSS). This is shown in Hsu and Murphy (1986).

$$\mathrm{BS} = \bar{y}(1 - \bar{y}) + \frac{1}{M}\sum_{k=1}^{K} M_k(f_k - \bar{y}_k)^2 - \frac{1}{M}\sum_{k=1}^{K} M_k(\bar{y}_k - y_k)^2 \tag{4.27}$$

M is the number of examples; K is the number of forecast bins (10); M_k is the number of examples in the k^{th} forecast bin; f_k is the mean forecast for the k^{th} bin; and $\bar{y}_k$ is the conditional event frequency for the k^{th} bin. BS ranges from $[0, 1]$, with lower values considered better. This equation is mathematically identical to Equation 4.15. The only difference is that Equation 4.27 is more convenient for the interpretation of the attributes diagram. An even more convenient form of the BS is given below.

$$\begin{cases} \mathrm{BS} = \mathrm{UNC} + \mathrm{REL} - \mathrm{RES} \\ \mathrm{UNC} = \bar{y}(1 - \bar{y}) \\ \mathrm{REL} = \frac{1}{M}\sum_{k=1}^{K} M_k(f_k - \bar{y}_k)^2 \\ \mathrm{RES} = \frac{1}{M}\sum_{k=1}^{K} M_k(\bar{y}_k - y_k)^2 \end{cases} \tag{4.28}$$

UNC is the uncertainty, ranging from $[0, 0.25]$; REL is the reliability, ranging from $[0, 1]$; and RES is the resolution, ranging from $[0, 1]$. These are three of the 10 aspects of forecast quality, defined conceptually in Table 4.1. Lower values of REL are considered better, and higher values of RES are considered better. However, no such evaluative assessments are made for UNC, since it depends only on climatology and is therefore out of human control.

In general, a "skill score" is the value of some verification statistic relative to a baseline. When using the BSS to evaluate the attributes diagram, the baseline is always climatology. The climatological BS is defined below.

$$\text{BS}^* = \bar{y}(1 - \bar{y}) + \frac{1}{M}\sum_{k=1}^{K} M_k(\bar{y}_k - \bar{y}_k)^2 - \frac{1}{M}\sum_{k=1}^{K} M_k(\bar{y}_k - \bar{y}_k)^2$$

$$\therefore \text{BS}^* = \bar{y}(1 - \bar{y}) = \text{UNC} \tag{4.29}$$

Thus, the BSS is as follows.

$$\text{BSS} = \frac{\text{BS}^* - \text{BS}}{\text{BS}^*} = \frac{\text{RES} - \text{REL}}{\text{UNC}} \tag{4.30}$$

Like any skill score, the BSS ranges from $(-\infty, 1]$, with higher values considered better. A positive BSS (RES $>$ REL) means that the model is better than climatology. This makes sense, because (a) lower values of REL are better, and higher values of RES are better; (b) a climatological forecast model always has RES = REL = 0. Thus, as long as RES $>$ REL, the model is better than climatology.

To our knowledge, unlike for AUC, there is no standard set of evaluative criteria for RES, REL, or BSS – except that a positive BSS is better than climatology.

The main advantage of the attributes diagram is that provides explicit information on the quality of probabilistic forecasts. Altogether, the joint PDF, ROC curve, performance diagram, and attributes diagram provide information on all 10 aspects of forecast quality (Table 4.1).

Chapter 5

Variable Selection

In general, there are two goals of variable selection: to find (a) an optimal set of variables for machine learning and (a) the most physically relevant variables for the problem. In our case the "problem" is predicting which storm cells will produce a wind gust ≥ 50 kt. These goals should be consistent with each other, but for practical reasons they are sometimes not. For example, if the most physically relevant variables are highly collinear, some learning algorithms (*e.g.*, those that depend on matrix inversion) might become unstable and perform poorly. Also, if there are too many physically relevant variables, the computational requirements might become too large for certain algorithms.

To satisfy these two goals, we use several methods and compare their results. The following subsections describe the methods used, which are split into two categories: wrapper methods and filter methods.

5.1 Wrapper Methods

The goal of a wrapper method (Kohavi and John, 1997) is to find the optimal set of variables for a specific machine-learning model (*e.g.*, logistic regression, feed-forward neural net, random forest, etc.). However, it is often impossible to explore the full hypothesis space (all combinations of variables). In general, if

there are N predictor variables, the number of possible combinations is $\sum_{j=1}^{N} \binom{N}{j}$. For $N = 431$, this number is $\binom{431}{1} + \binom{431}{2} + \ldots + \binom{431}{431} = 5.5 \times 10^{129}$. Even if each combination could be explored in 10^{-100} seconds, this procedure would take longer than the age of the universe. Thus, wrapper methods are often limited to finding a *near*-optimal set of variables.

The advantage of wrapper methods is that, because they explicitly optimize model performance, they usually lead to better predictions than filter methods. However, because wrapper methods depend on a specific model, their results are less widely applicable. Listed below are three reasons that results of a wrapper method may have limited application.

1. If a wrapper method is used with a model that cannot handle many variables (*e.g.*, logistic regression), the selected variables may exclude some that are still physically relevant.
2. If a wrapper method is used with a model that performs built-in variable selection (*e.g.*, decision trees), the selected variables may include some that have little physical relevance.
3. In both cases, although selected variables may lead to near-optimal performance of the given model, they will probably lead to highly suboptimal performance of other learning models.

These shortcomings can be addressed by using filter methods, as well as wrapper methods with different underlying models, then comparing results.

For all wrapper methods, storm objects are sampled as described in Section 4.2: from the uniform distribution to train the base model, from the best-observed distribution to train the calibration model (isotonic regression), and from the best-observed distribution to test the combined model (base model + isotonic regression).

5.1.1 Sequential Forward Selection

Sequential forward selection (SFS) (Aha and Bankert, 1996) is a general wrapper method, which means that it can be used with any learning model. The procedure is described in Algorithm 5.1 and Figure 5.1.

SFS begins with a constant model, which predicts the mean label of all training examples (this is known as a "climatology forecast"). Then, at each selection step i, SFS begins with a model containing i - 1 variables and finds the best model containing one additional variable. In other words, SFS performs a greedy search at each selection step, adding the best of the remaining N - (i - 1) variables to the model. SFS stops when either (a) model performance has not improved in the last k selection steps (k is `latencyTime` in Algorithm 5.1) or (b) all N variables have been added. In our experience (thousands of runs with $k = 5$), case (b) has never occurred.

The main advantage of SFS is that, by not allowing backtracking (*i.e.*, not allowing a variable to be deselected), it reduces the number of combinations tested to $\frac{1}{2}N(N-1)$, which is only 92,665 when $N = 431$. However, this is still intractable for more sophisticated learning models such as FFNN, random forests, and GBT ensembles.

We use two learning algorithms with SFS: logistic regression as the base model and isotonic regression for probability calibration. We use logistic regression because it is fast, does not perform built-in variable selection (which would invalidate the results of SFS), and is well suited for binary classification (Section 4.1.1.1). We use isotonic regression because it is more appropriate than Platt scaling when the base model is logistic regression (see end of Section 4.1.2.1).

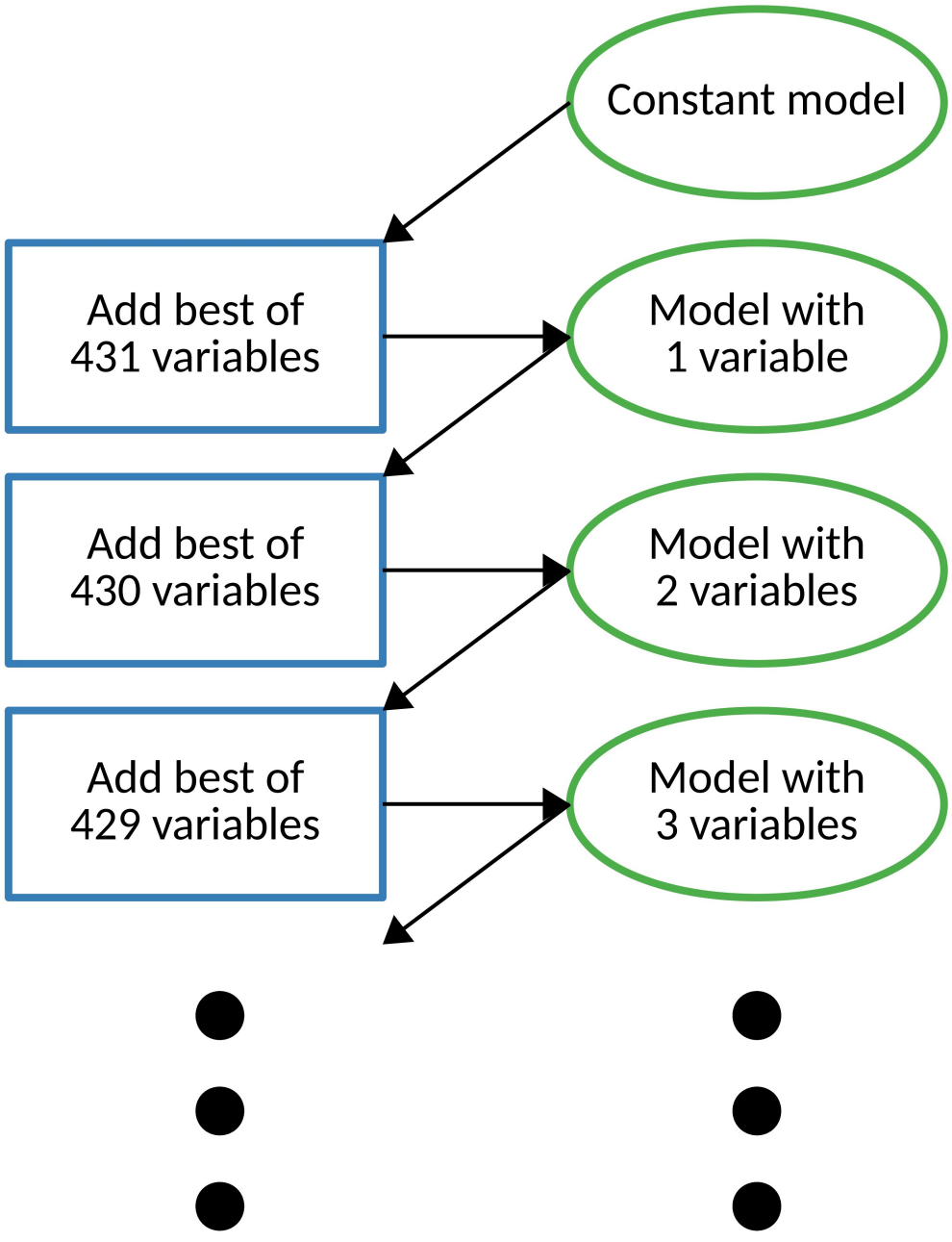

Figure 5.1: Schematic for sequential forward selection. Each blue box is an action, and the green ellipse to the right is the resulting model. Black (grammatical) ellipses indicate that the process may continue.

Algorithm 5.1: Sequential forward selection.

Input: `trainExamples` = trng examples, each with N predictors and label

`testExamples` = testing examples; `predictors` = list of N predictors

`verifStat` = verification stat (used to quantify model performance)

`latencyTime` = stop if `verifStat` not improved over this many selection steps

Output: `bestPredictors` = list of predictors selected for final model

```
// bestPerformance starts at index 0; other arrays start at 1.
model = constant model (predicts mean label of all trainExamples);
bestPerformance(0) = verifStat(model(testExamples));

for i = 1...length(predictors) do
    testPredictors = all predictors not in bestPredictors;
    testPerformance = empty set;
    for j = 1...length(testPredictors) do
        trainPredictors = bestPredictors ∩ testPredictors(j);
        model = model trained with trainExamples.trainPredictors;
        testPerformance(j) = verifStat(model(testExamples));
    end

    // If smaller verifStat is better, change "max" to "min".
    (bestPerformance(i), maxIndex) = max(testPerformance);
    bestPredictors(i) = testPredictors(maxIndex);

    // If smaller verifStat is better, change last operator to ≥.
    if i ≥ latencyTime ∩ bestPerformance(i) ≤ bestPerformance(i -
     latencyTime) then
        return bestPredictors(1...(i - latencyTime));
    end
end
```

5.1.2 Sequential Backward Selection

Sequential backward selection (SBS) (Aha and Bankert, 1996) is also a general wrapper method, so can be used with any type of learning model. The main difference between this and SFS is that SBS serially eliminates variables, rather than adding them. The procedure is described in Algorithm 5.2 and Figure 5.2.

At each selection step i, SBS begins with a model containing N - (i - 1) variables and finds the best model containing one fewer variable. In other words, SBS performs a greedy search at each selection step, removing the worst of N - (i - 1) variables from the model. SBS stops when either (a) model performance has not improved in the last k selection steps (k is `latencyTime` in Algorithm 5.2) or (b) all N variables have been removed, yielding a constant model.

Because SFS and SBS work in opposite directions, they usually give at least slightly different answers, which can be compared. When a variable is selected by both methods, one can be fairly certain that it (a) is physically relevant and (b) will lead to good performance for other learning models.

The main disadvantage of SBS is that, when N is large, it requires much more computing time than SFS. This is because the first few selection steps in SFS involve training a model with only a few variables, whereas the first few steps in SBS involve training a model with $\sim N$ variables. Thus, if both algorithms stop well before N selection steps (which in our experience is always the case), the average number of variables per SBS-trained model is much greater than per SFS-trained model. Also, simple learning models (which are required for SFS and SBS, due to runtime issues discussed in Section 5.1.1) tend to be unstable with a large number of variables, so removing one variable x_j often leads to very different performance, even if x_j is not physically relevant. Thus, when N is large SFS and SBS usually give very different results, which does not allow for a useful comparison.

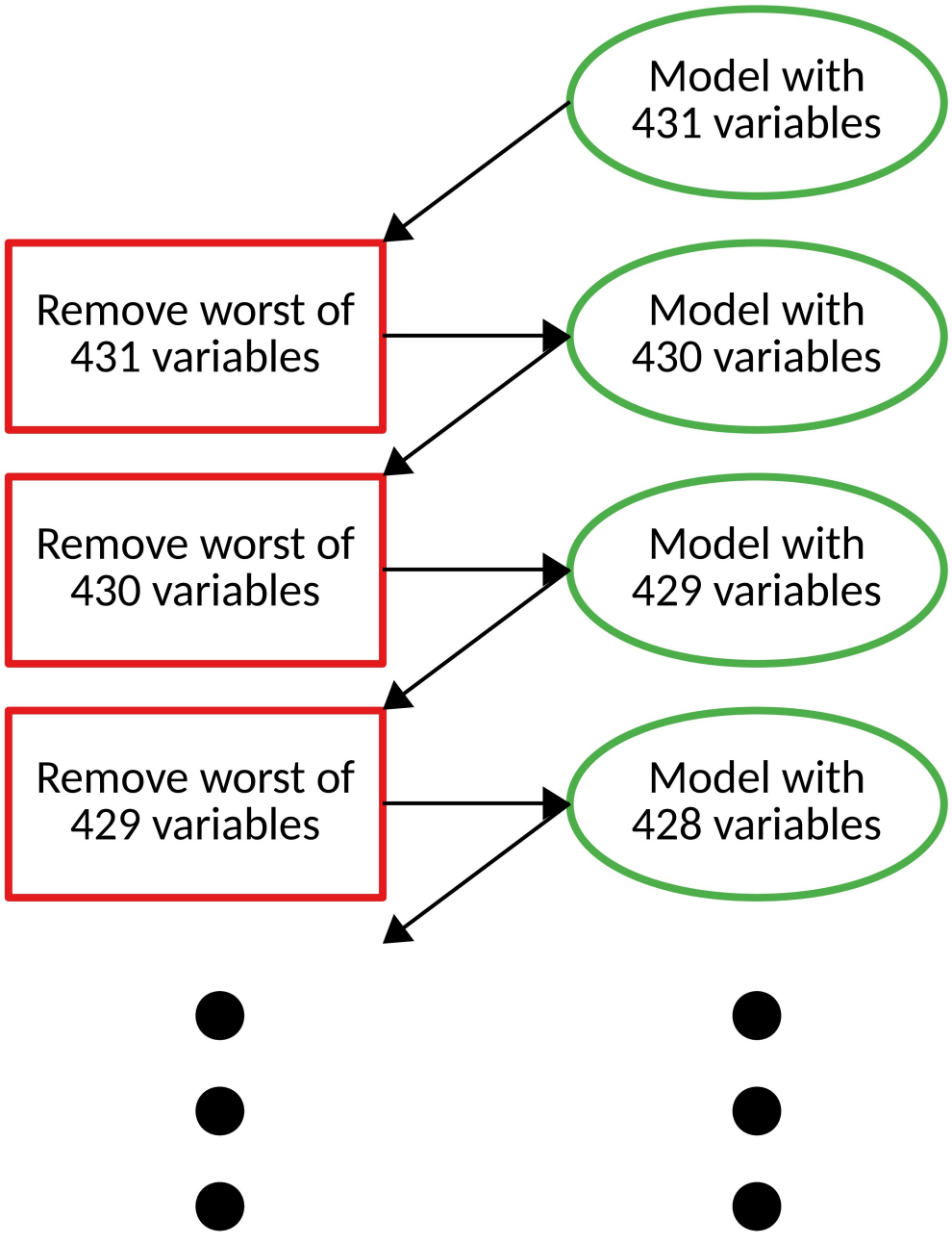

Figure 5.2: Schematic for sequential backward selection. Each red box is an action, and the green ellipse to the right is the resulting model. Black (grammatical) ellipses indicate that the process may continue.

Algorithm 5.2: Sequential backward selection.

Input: same as Algorithm 5.1

Output: same as Algorithm 5.1

```
// worstPerformance starts at index 0; other arrays start at 1.
model = model trained with all predictors;
worstPerformance(0) = verifStat(model(testExamples));
worstPredictors = empty set;

for i = 1...length(predictors) do
    testPredictors = all predictors not in worstPredictors;
    testPerformance = empty set;
    for j = 1...length(testPredictors) do
        trainPredictors = all testPredictors except testPredictors(j);
        model = model trained with trainExamples.trainPredictors;
        testPerformance(j) = verifStat(model(testExamples));
    end

    // If smaller verifStat is better, change "min" to "max".
    (worstPerformance(i), minIndex) = min(testPerformance);
    worstPredictors(i) = worstPredictors(maxIndex);

    // If smaller verifStat is better, change last operator to ≤.
    if i ≥ latencyTime ∩ worstPerformance(i) ≥ worstPerformance(i -
     latencyTime) then
        return bestPredictors = all predictors except
         worstPredictors(1...(i - latencyTime));
    end
end

return bestPredictors = empty set;
```

5.1.3 Decision-tree Method

The decision-tree method (DTM) may be used for a single decision tree, random forest, or GBT ensemble. For a single decision tree, the importance of variable x_j is the average reduction in deviance over all splits involving x_j. For a random forest or GBT ensemble, the importance of x_j is the same, except that splits involving x_j may occur in multiple trees. The "reduction in deviance" is simply deviance at the parent node minus summed deviances at the child nodes, as defined below. D comes from Equation 4.2.

$$\Delta D = D_{parent} - D_{left} - D_{right} \tag{5.1}$$

Variable importance (VI) is defined below.

$$\text{VI}(x_j) = \frac{\sum\limits_{s \in \text{splits on } x_j} \Delta D_s}{\sum\limits_{s \in \text{splits on } x_j} 1} \tag{5.2}$$

The main advantage of the DTM is that, unlike SFS and SBS, it can be run quickly for random forests and GBT ensembles. This is important, because random forests and GBT ensembles are usually the best models (Section 6.1.3) and the credibility of variable importance increases with model quality. The main disadvantage of the DTM is that it cannot be used with other learning models.

5.2 Filter Methods

Filter methods select predictor variables based on how well they discriminate between values of the dependent variable (Kohavi and John, 1997). For binary classification, filter models select variables based on how well they discriminate between labels of 0 and 1. If the class-conditional PDFs of x_j ($p(x_j \mid y = 0)$ and

$p(x_j \mid y = 1)$) are very similar, x_j has low discrimination between the two classes. If the PDFs are very different, x_j has high discrimination.

The advantage of filter methods is that, because they do not depend on a specific learning model, their results are more widely applicable. In other words, variables selected by a filter method should lead to satisfactory performance for many learning models. Also, because filter methods are not affected by the peculiarities of a specific learning model (*e.g.*, an inability to deal with many variables or collinear variables), their results may be more appropriate for physical interpretation.

For all filter methods, storm objects are sampled from the best-observed distribution (Section 4.2.2), which is most representative of the real world.

5.2.1 *J*-measures

The J-measure is the Kullback-Leibler divergence (Jeffreys, 1946; Lin, 1991) between probability distributions $p(x_j \mid y = 0)$ and $p(x_j \mid y = 1)$. See Figure 5.3. For a discrete x_j, the J-measure is defined below, where K is the number of possible values for x_j.

$$J(x_j) = \sum_{k=1}^{K} [p(x_j = x_{jk} \mid y = 0) - p(x_j = x_{jk} \mid y = 1)] \log_2 \left[\frac{p(x_j = x_{jk} \mid y = 0)}{p(x_j = x_{jk} \mid y = 1)} \right] \tag{5.3}$$

If x_j is continuous (like all of our predictor variables), it is discretized into K bins, where X_{jk} is the range of values in the k^{th} bin.

$$J(x_j) = \sum_{k=1}^{K} [p(x_j \in X_{jk} \mid y = 0) - p(x_j \in X_{jk} \mid y = 1)] \log_2 \left[\frac{p(x_j \in X_{jk} \mid y = 0)}{p(x_j \in X_{jk} \mid y = 1)} \right] \tag{5.4}$$

The J-measure ranges from $[0, \infty)$. A higher J-measure means that there is more difference between the class-conditional PDFs ($p(x_j \mid y = 0)$ and $p(x_j \mid$

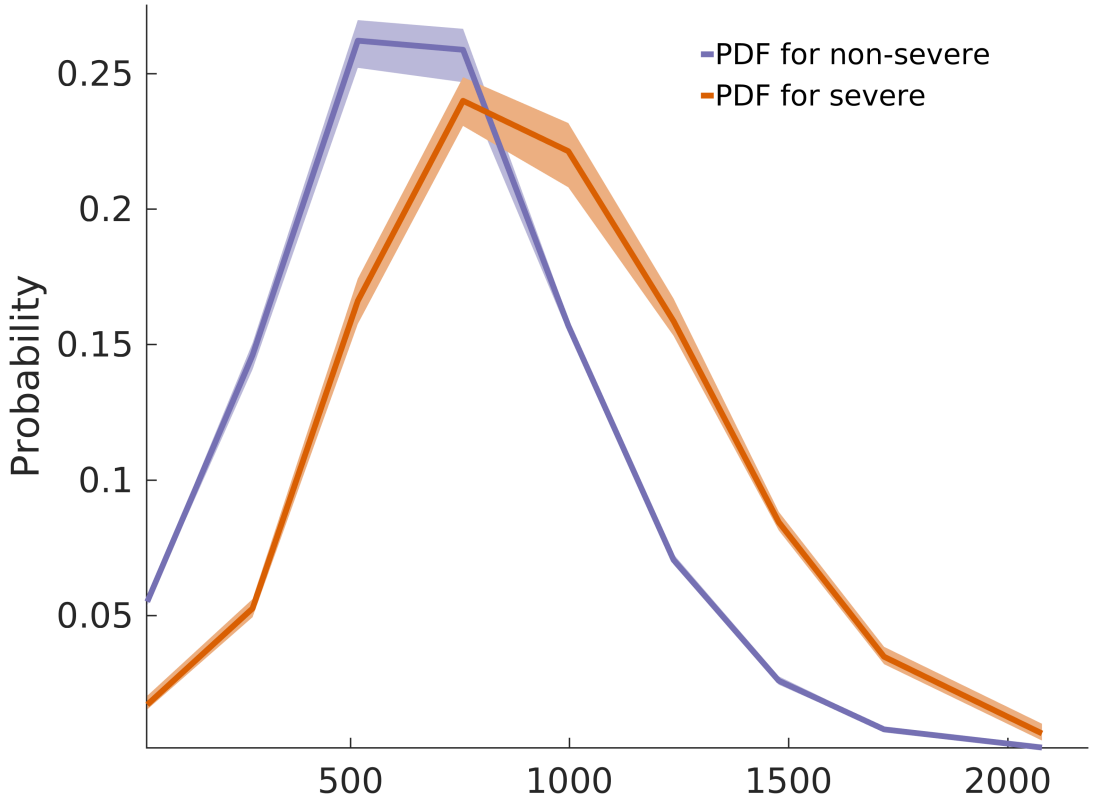

Figure 5.3: PDFs used to calculate J-measure for a single variable. The orange curve is the PDF for storm objects with label = 1; the purple curve is for storm objects with label = 0. Each solid line is a mean, and each shaded area is a 95% confidence interval, determined by bootstrapping.

$y = 1)$), which means that x_j is a better discriminator between severe and non-severe-wind-producing storms. A J-measure of zero means that x_j has absolutely no discrimination between severe and non-severe-wind-producing storms.

5.3 From Variable-ranking to Selection

Two of the four methods in Sections 5.1-5.2 (decision trees and J-measures) only rank variable importance, rather than selecting variables. We consider two ways to use these rankings for explicit selection. The linear-dependence-controlling (LDC) method is used only for J-measures, which have no other way to account

for collinearity, because they consider only one variable at a time. The LDC would be inappropriate for ranking variable importances from the decision-tree (or any other wrapper) method, because the point of a wrapper method is to find variables that optimize the performance of a learning model, regardless of whether or not these variables are collinear. The linear-dependence-agnostic (LDA) method can be used for any set of variable importances, because it simply selects the top-ranked variables with no further processing.

5.3.1 Linear-dependence-agnostic Method

Variables may be selected by one of three criteria (letting VI be variable importance, as in Section 5.1.3):

1. Choose the top N^* variables.
2. Choose all variables with VI $\geq$ VI*.
3. Keep choosing from the top, until the relative difference in VI between two successively ranked variables exceeds some threshold. This criterion is stated mathematically in Equation 5.5, where VI(x_k) is the importance of the k^{th}-ranked variable and r is the aforementioned threshold.

$$\frac{\text{VI}(x_j) - \text{VI}(x_{j+1})}{\text{VI}(x_j)} \geq r \tag{5.5}$$

The advantage of the LDA method is that many collinear variables may be selected, which may aid physical insight into the problem. For example, if 0–3-km, 3–6-km, 850–500-mb, and 700–500-mb lapse rates are selected, this indicates that both low- and mid-level lapse rates are important for severe wind. If collinear variables were eliminated, perhaps only one lapse rate would be returned, which would obfuscate this fact. Disadvantages of the LDA method are that (a) too many collinear variables might be returned, thus overwhelming the user and hindering physical interpretation; (b) some learning methods perform

poorly with many collinear variables; and (c) it is highly subjective, since both a criterion and parameter ($N*$, VI*, or r) must be chosen. The parameter should be chosen in a way that gives a manageable number of variables (usually on the order of 10), but to our knowledge there are no widely accepted values for any of these parameters.

5.3.2 Linear-dependence-controlling Method

Variables are selected by the procedure described in Algorithm 5.3. Each time that a variable x_j is selected, all variables x_k meeting the following criteria are eliminated:

1. $\text{VI}(x_k) < \text{VI}(x_j)$ at confidence level α.
2. $|\text{Pearson}(x_j, x_k)| \geq r*$ at confidence level α, where $|\text{Pearson}(x_j, x_k)|$ is the absolute Pearson correlation and r^* is a threshold.

The confidence interval for each variable importance and absolute Pearson correlation may be determined by bootstrapping.

The main advantage of the LDC method is that it is less subjective than LDA. For the LDA method, one must choose from three selection criteria, then choose the parameter that goes with said criterion. For the LDA method, although two parameters must be chosen, there are widely accepted values for a strong confidence interval (95%, leading to $\alpha = 0.05$ for the one-tailed hypothesis tests in steps 1 and 2) and moderate (0.3) or strong (0.6) correlation.

Algorithm 5.3: Linear-dependence-controlling variable selection. CI = confidence interval; VI = variable importance.

Input: `predictors` = list of N predictors

`impDiff(i, j)` = bottom of CI for VI(`predictors(i)`) - VI(`predictors(j)`)

`absCorr(i, j)` = bottom of CI for |Pearson(`predictors(i)`, `predictors(j)`)|

`meanImp(i)` = mean importance (VI) of `predictors(i)`

`r*` = threshold for absolute Pearson correlation

Output: `bestPredictors` = list of predictors selected

```
bestPredictors = empty set;
while predictors is not empty do
    // Let i = index of best remaining predictor.
    (maxImportance, i) = max(meanImp);
    Append predictors(i) to bestPredictors;
    // Remove if significantly correlated with, and significantly
       less important than, predictors(i).
    for j = 1...length(predictors) do
        if impDiff(i, j) > 0 ∩ absCorr(i, j) > r* ∩ i ≠ j then
            Remove predictors(j) from predictors;
            Remove corresponding entries from impDiff, absCorr, meanImp;
            j = j - 1 ;              // So that entries are not skipped.
        end
    end
    Remove predictors(i) from predictors;
    Remove corresponding entries from impDiff, absCorr, meanImp;
end
return bestPredictors;
```

Chapter 6

Experiments

6.1 Experiment 1: Comparison of Machine-learning Algorithms

We compare the performance of five base-model algorithms: logistic regression (described in Section 4.1.1.1), LREN (Section 4.1.1.2), FFNN (Section 4.1.1.3), random forests (Section 4.1.1.5), and GBT ensembles (Section 4.1.1.6). Isotonic regression (Section 4.1.2.2) is used to calibrate probabilities for all base models. We ruled out Platt scaling *a priori*, because (a) of the five base-model algorithms, Platt scaling is appropriate only for GBT ensembles (see end of Section 4.1.2.1); (b) even for GBT ensembles, Platt scaling does not appear to perform better than isotonic regression (Figures 2-3 of Niculescu-Mizil and Caruana, 2005). The five base-model algorithms are compared for each buffer distance (0, 5, and 10 km) and lead-time window ($[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes).

Sampling techniques are as described in Section 4.2. Storm objects are drawn from a uniform distribution of U_{max} for training the base model, from the best-observed distribution (with $N^*_{obs} = 25$) for training the isotonic model, and from the best-observed distribution (again with $N^*_{obs} = 25$) for testing the combined model (base model plus isotonic regression). 50% of available examples (those

drawn from the uniform distribution) are used to train the base model; 75% of available examples (those from the best-observed distribution that are independent of base-model-training data) are used to train the isotonic model; and remaining examples (those from the best-observed distribution that are independent of both training sets) are used to test the combined model. All discussions of "independence" in this chapter use the 24-hour criterion defined in Section 4.2.4.

We set $N^*_{obs} = 25$ for the best-observed distribution, which means that all storm objects with $U_{max} \geq 50$ kt or $N_{obs} \geq 25$ are used. We found that lower values allow many more negative cases (storm objects with label = 0, which may actually have produced severe winds that were unobserved), leading to a sharp decrease in forecast resolution (an inability to forecast probabilities near 100%), even after calibration.

Model selection is described in Section 6.1.1; cross-validation methods are described in Section 6.1.2; and the parameters used for each model, as well as details of its execution, are given in Appendix A. The procedures described in all subsections are repeated for each buffer distance and lead-time window. Figure 6.1 shows how these procedures fit together.

6.1.1 Model Selection

Each base model is trained with all combinations of parameters listed in Appendix A, then calibrated with isotonic regression. This leads to 1786 combined models: one with logistic regression as the base model, 25 for LREN, 800 for FFNN, 210 for random forests, and 750 for GBT ensembles.

Combined models are ranked by area under the ROC curve (AUC) (Section 4.3.4). We also considered using maximum critical success index (CSI) (Section 4.3.5) and Brier skill score (BSS) (Section 4.3.6). We decided against maximum

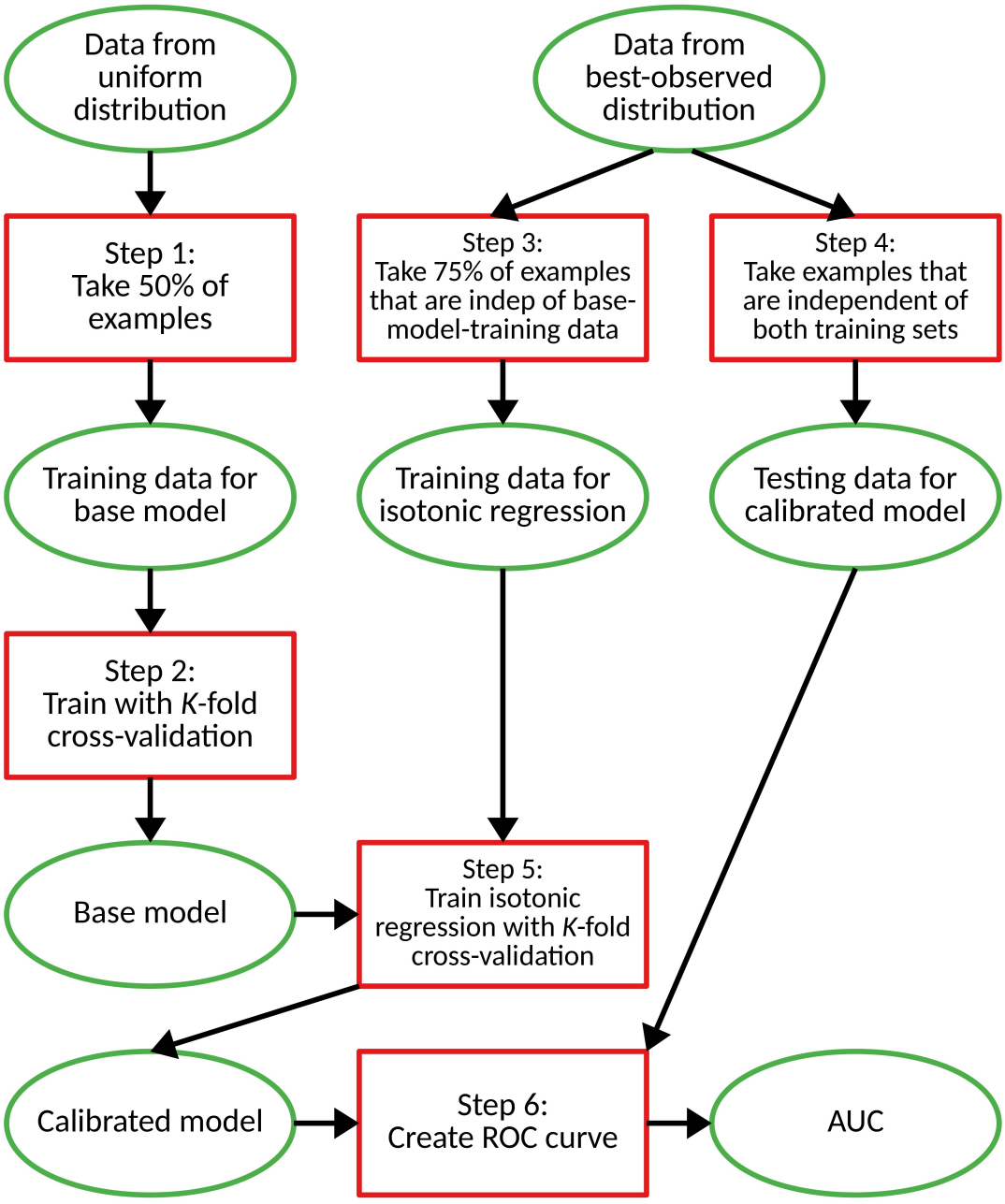

Figure 6.1: Flow chart for Experiment 1. Each red box is an action, and each green ellipse is an object (or set of objects). Steps 1-6 are repeated for each buffer distance, lead-time window, and each of 1786 sets of base-model parameters.

CSI because it is an unstable statistic: it considers only one probability threshold (f^* in Equation 4.21), whereas AUC considers all probability thresholds. Thus, it is easier to obtain a spuriously high maximum CSI than a spuriously high AUC. We decided against the BSS because we found that, even when two models have very different AUC and maximum CSI, they often have very similar BSS. Furthermore, we found that models with the best AUC tend to have nearly the best maximum CSI and BSS.

Ideally, we would use the following method to select combined models.

1. Use the same training set A for all base models and B for all isotonic models, so that the same testing set C can be used for all combined models.
2. Bootstrap the testing set K times. For each model, find the AUC for each bootstrapped testing set and the mean AUC over all bootstrapped sets.
3. Let the model with the highest mean AUC be F_1. For all other models F, find $\text{AUC}(F_1) - \text{AUC}(F)$ for all bootstrapped testing sets. Then find the 5^{th} percentile of $\text{AUC}(F_1) - \text{AUC}(F)$ and call this $(\Delta\text{AUC})_5$.
4. Select all models with $(\Delta\text{AUC})_5 \leq 0$ (for which one cannot say with 95% confidence that F_1 is better).

However, most of our testing sets (one for each buffer distance and lead-time window) contain ~10^4, for which the creation of ROC curves takes a lot of computing time. Thus, it was impractical to bootstrap the AUC many times for each model. In previous experiments with bootstrapping (involving many fewer models), we found that when the non-bootstrapped AUC (computed for all testing data) differs by ≤ 0.01 between F and F_1, $(\Delta\text{AUC})_5 \leq 0$. In other words, when the non-bootstrapped AUC differs by ≤ 0.01, the difference is not statistically significant. Thus, we amended the above procedure as follows.

1. Find the non-bootstrapped AUC for each model.

Algorithm	Number of Folds	Independent Folds?	Criterion
Logistic regression	10	Yes	AUC
LREN	2	Yes	AUC
FFNN	2	Yes	AUC
Random forests	2	No	AUC
GBT ensembles	2	No	AUC
Isotonic regression	10	Yes	Brier score

Table 6.1: Cross-validation parameters for each machine-learning algorithm.

2. Let the highest AUC be AUC_{max}. Keep all models with $\text{AUC} \geq \text{AUC}_{max} - 0.01$.

6.1.2 Cross-validation

Cross-validation methods are listed in Table 6.1. Ten-fold cross-validation is preferred, but two-fold is used when ten-fold would be too computationally intensive. In general, for K-fold cross-validation, training examples are split into K mutually exclusive sets called "folds". Then K models are trained, where the k^{th} model is validated with the k^{th} fold and trained with all others. The separation of training and validation examples for each model is called a "partition". Each model is verified on its respective validation fold, which allows the best model to be kept and the others to be discarded. As long as the K folds are mutually independent (addressed later in this section) and testing data are independent of the training/validation data (addressed in Section 6.1, where the training/validation data are just called the "training" data), the best model on its respective validation fold should be the best model on the testing set.

For base models, we keep the model with the highest AUC on its respective validation fold. AUC is used for reasons discussed in Section 6.1.1. For isotonic regression, we keep the model with the lowest Brier score (BS) (Equation 4.15) on its respective validation fold. This is because (a) BS is the objective function for isotonic regression (Section 4.1.2.2) and (b) often, when isotonic regression causes a large decrease in BS (improvement in probability calibration), it causes a slight decrease (worsening) in AUC. We are willing to accept this trade for a well calibrated model. Unlike in Section 6.1.1, we do not test whether this AUC is significantly higher (or BS is significantly lower) than those for the other partitions. This is because, for a given set of model parameters, we do not need to know which partition leads (or partitions lead) to the best model. We are concerned only with finding and keeping the best model.

For logistic regression, LREN, FFNN, and isotonic regression, we ensure mutual independence among the K folds. However, for random forests and GBT ensembles, (a) cross-validation is built into the MATLAB function; (b) cross-validation performs a crucial role in the MATLAB function; and (c) the examples in each fold cannot be specified. Thus, for random forests and GBT ensembles, we cannot make the training and validation data fully independent. We accept that this may lead to the wrong cross-validated model being kept. However, we still ensure that the testing data are independent of both training and validation data, so this should have little effect on model selection.

6.1.3 Results

Selected models are shown in Appendix B. For a given buffer distance and lead-time window, if more than 20 models have been selected, only the top 20 are shown. We make two key observations from Appendix B, the first of which is that most of the base models selected are either random forests or GBT ensembles

Buffer Dist	**Lead Time**	**# Tree Ensembles**	**# Selected Models**
0 km	$[0, 15]$ min	40	72
0 km	$[15, 30]$ min	35	39
0 km	$[30, 45]$ min	48	55
0 km	$[45, 60]$ min	17	17
0 km	$[60, 90]$ min	7	7
5 km	$[0, 15]$ min	15	19
5 km	$[15, 30]$ min	34	41
5 km	$[30, 45]$ min	89	120
5 km	$[45, 60]$ min	38	52
5 km	$[60, 90]$ min	13	18
10 km	$[0, 15]$ min	27	35
10 km	$[15, 30]$ min	28	42
10 km	$[30, 45]$ min	18	22
10 km	$[45, 60]$ min	11	17
10 km	$[60, 90]$ min	7	7

Table 6.2: Number of tree ensembles selected for each buffer distance and lead-time window.

(Table 6.2). This was expected, given (a) the ability of decision trees (and ensembles thereof) to deal with a large number of predictor variables (Section 4.1.1.4) and (b) past successes of GBT ensembles in meteorology [*e.g.*, the top three finishers in a recent contest to predict surface incoming solar radiation were all GBT ensembles (McGovern et al., 2015)].

Our second observation from Appendix B is that there is very little consistency in the parameters for selected models, which may suggest one of two things: (a) the parameters have little effect on performance in general or (b) the parameters have little effect within the range of values tested. The second explanation is more likely, since we worked extensively with the five base-model algorithms before Experiment 1 and had roughly narrowed down a range of optimal values for each. Thus, in Experiment 1 we tested only values within these ranges, which were likely to be more fruitful than values outside of said ranges.

For each buffer distance and lead-time window, Figures 6.2-6.6 show the ROC curve, performance diagram, and reliability curve for the best model. These figures are created by applying the combined model (base model + isotonic regression) to the testing data (which are independent of both the base-model-training and isotonic-training data). The attributes diagram for each buffer distance and lead-time window is shown in Figures 6.7-6.11. We could not plot more than one attributes diagram on the same axes, because the attributes diagram includes four special lines and polygons – the climatology line, no-resolution line, no-skill line, and positive-skill area (Section 4.3.6) – which vary for each buffer distance and lead-time window.

Observations from Figures 6.2-6.11 are listed below.

1. For each buffer distance, the AUC, maximum CSI, and BSS decrease (worsen) with lead time; for each lead-time window, the same values decrease with buffer distance (Figures 6.2-6.6). Both of these relationships

make sense, since physical phenomena are generally more difficult to predict when they are more spatially and temporally distant.

2. The vast majority of each reliability curve (Figures 6.2-6.6) lies within the positive-skill area (shaded in blue). This means that for most combinations of buffer distance, lead-time window, and forecast probability, our models have a positive BSS – *i.e.*, they have a lower (better) Brier score (Equation 4.15) than a climatological forecast.
3. For each buffer distance, the climatology line in the attributes diagram (Figures 6.7-6.11) moves to the left with lead time. This means that the frequency of the event (maximum storm-object wind $U_{max} \geq 50$ kt) decreases with lead time, as expected. Thunderstorms are short-lived phenomena, and as lead time increases, it becomes more likely that the storm will be either weaker or non-existent (and therefore incapable of producing severe winds).

   However, for each lead-time window, the climatology line also moves to the left with buffer distance. This means that the frequency of the event ($U_{max} \geq 50$ kt) decreases with buffer distance, which is counterintuitive. Increasing the buffer distance around the storm object can only increase U_{max}, because this increases the area from which wind observations are linked to the storm object. This paradox is explained by the way that data are sampled for model calibration and testing (Section 4.2.2). For each buffer distance and lead-time window, we use only storm objects with $N_{obs} \geq 25$ associated wind observations or $U_{max} \geq 50$ kt (after ensuring that the percentage of dead storms in the sampled dataset equals that in the full dataset). As buffer distance increases, the number of storm objects with $N_{obs} \geq 25$ wind observations increases, and this apparently increases at a faster rate than the number with $U_{max} \geq 50$ kt. Thus, increasing the

buffer distance allows for more "well observed" storm objects with $U_{max} <$ 50 kt, which decreases the event frequency.

4. As the climatology line in the attributes diagram moves to the left (with increasing buffer distance or lead-time window), the forecast histogram (right-hand side of Figures 6.7-6.11) becomes more left-peaked. In other words, as the event becomes rarer, high forecast probabilities become rarer and low forecast probabilities become more common. This is an encouraging result.
5. The AUC values (Figures 6.2-6.6) for 14 of 15 pairs of buffer distance and lead-time window are ≥ 0.9, which is considered "excellent" by the criteria listed in Section 4.3.4. Only for the greatest buffer distance (10 km) and lead-time window ($[60, 90]$ minutes) is the AUC < 0.9. Here the AUC is 0.89, which is still considered "good".
6. As mentioned in Sections 4.3.5 and 4.3.6, there is no standard set of evaluative criteria for maximum CSI or BSS. However, points 2, 4, and 5 suggest that our models have achieved very good performance.
7. For each buffer distance and lead-time window, the maximum CSI (Figures 6.2-6.6) occurs with a frequency bias (Equation 4.23a) of ~1, which is ideal. This means that the event is forecast nearly as often as it occurs.

6.2 Experiment 2: Variable Selection

Three methods are used for each buffer distance and lead-time window: sequential forward selection (SFS) (Section 5.1.1), the decision-tree method (DTM) (Section 5.1.3), and J-measures (Section 5.2.1). Sequential backward selection (SBS) is omitted for reasons discussed in Section 5.1.2 (mainly that it takes too much computing time).

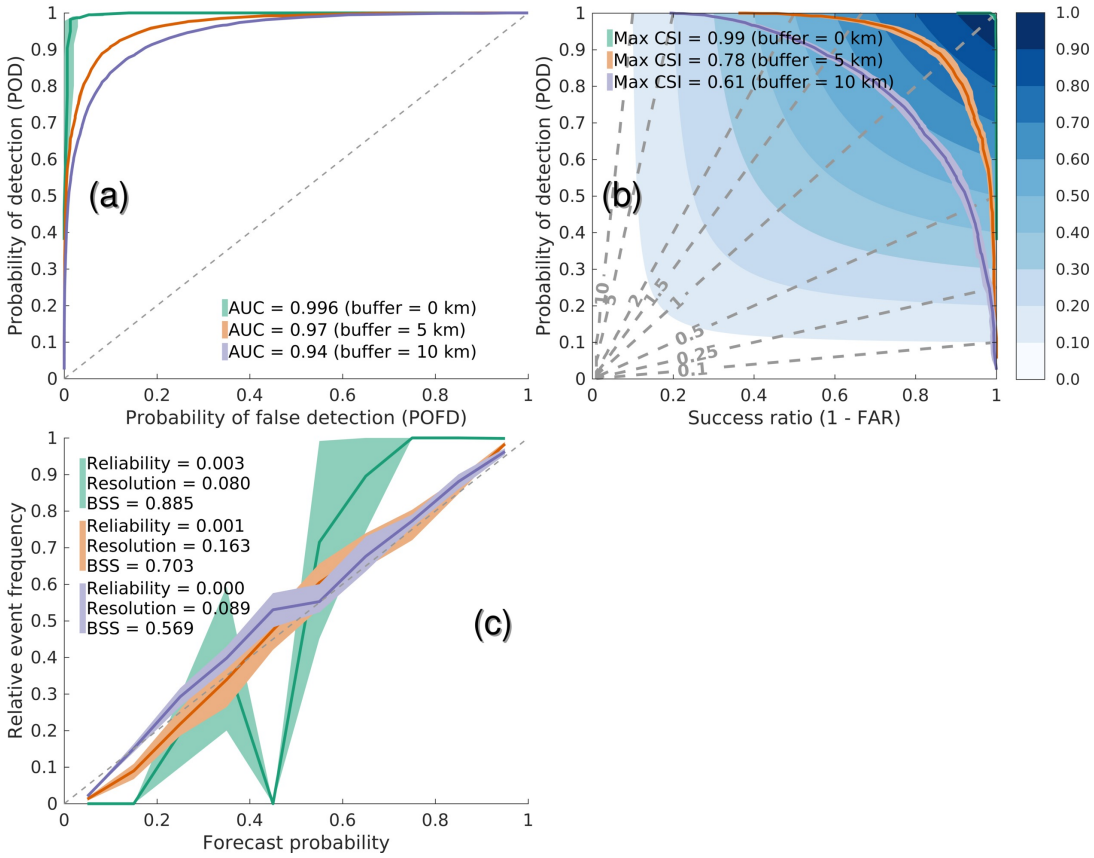

Figure 6.2: (a) ROC curves, (b) performance diagram, and (c) reliability curves for best models with lead time of $[0, 15]$ minutes. Each solid curve is a mean, and each shaded area is a 95% confidence interval, over 25 bootstrap replicates of the testing data. Base model for 0-km buffer is a random forest; base model for 5-km and 10-km buffers is a GBT ensemble.

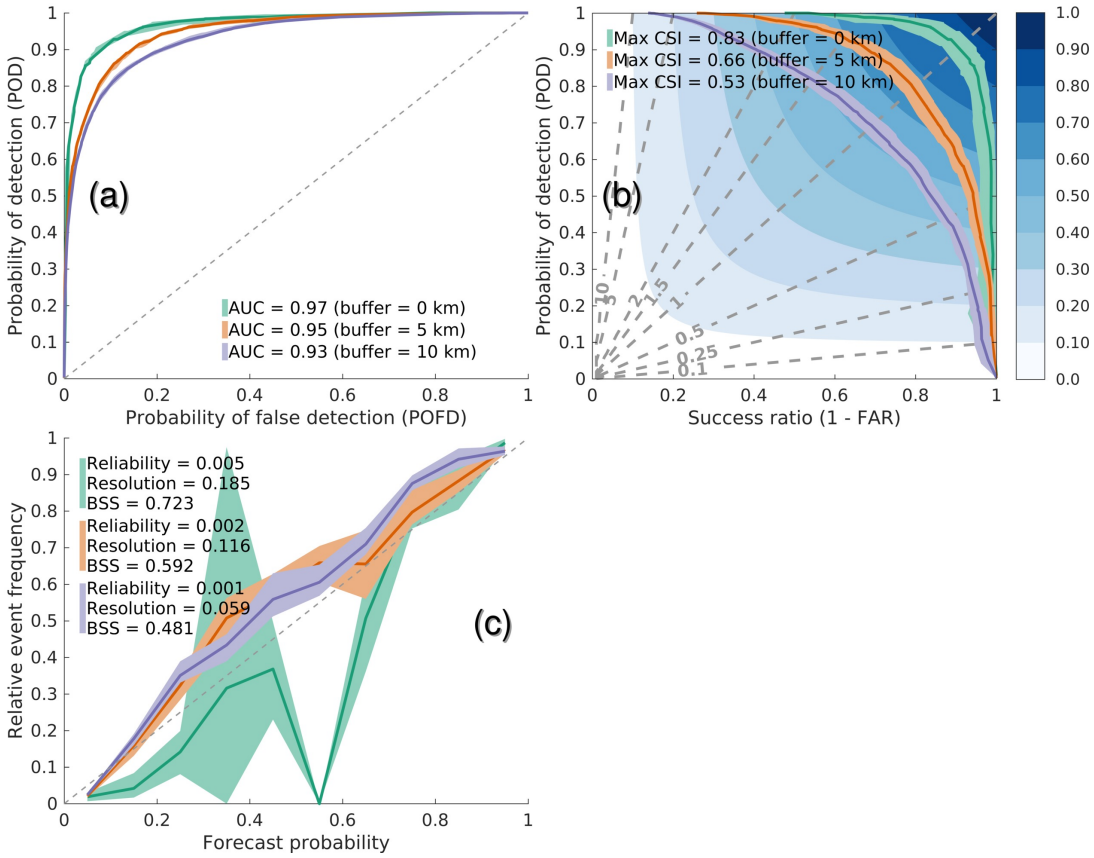

Figure 6.3: As in Figure 6.2, except for a lead time of $[15, 30]$ minutes. Base model for 0-km and 5-km buffers is a GBT ensemble; base model for 10-km buffer is LREN.

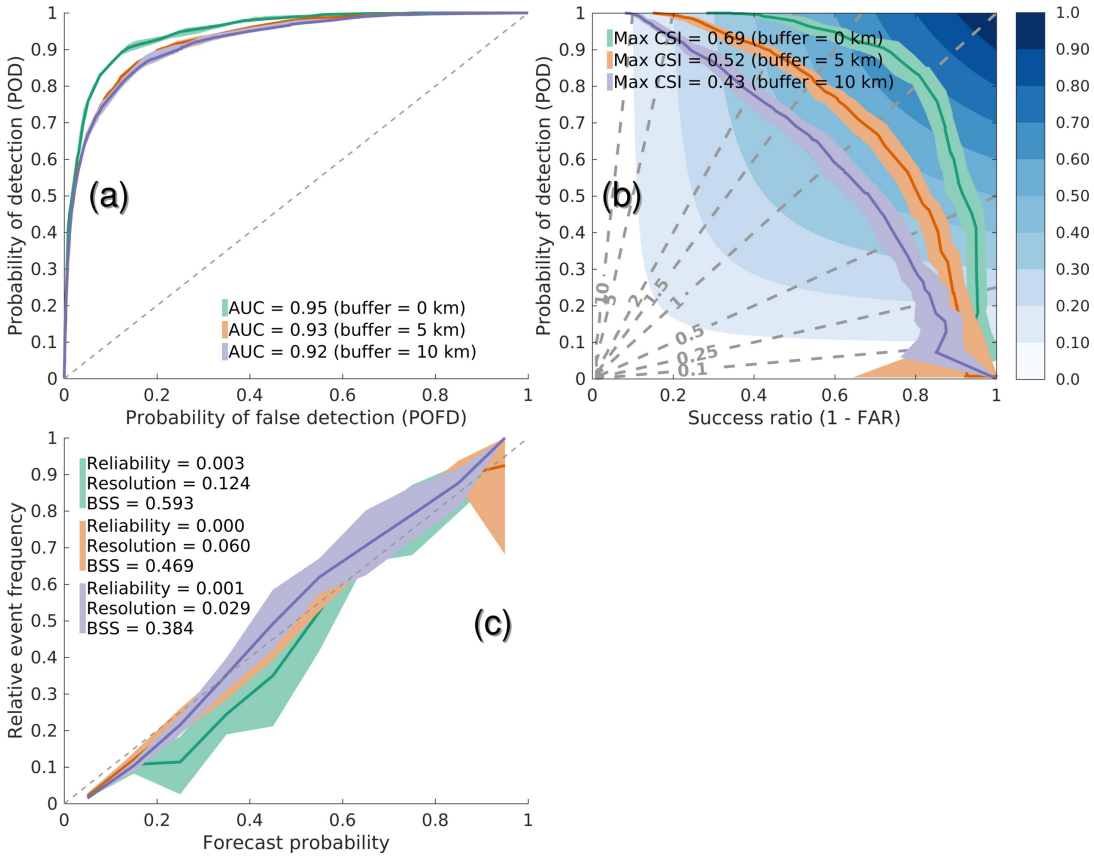

Figure 6.4: As in Figure 6.2, except for a lead time of $[30, 45]$ minutes. Base model for 0-km buffer is a random forest; base model for 5-km and 10-km buffers is an FFNN.

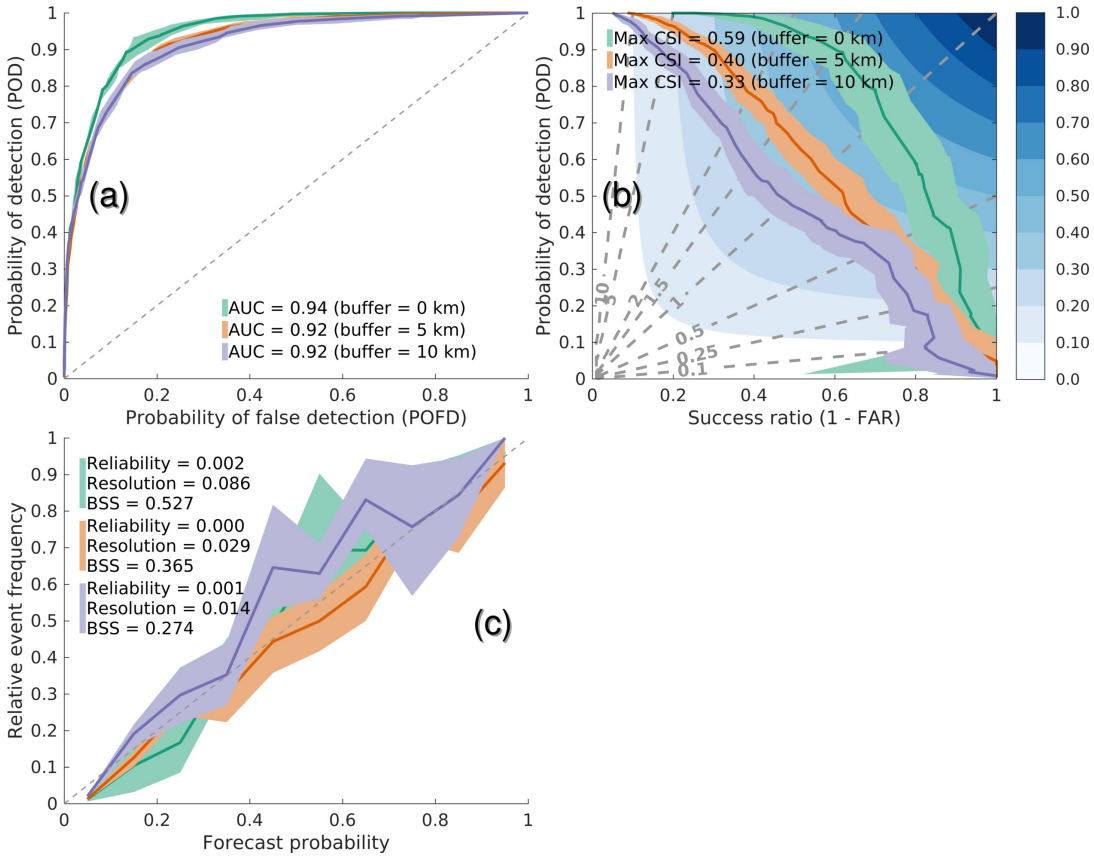

Figure 6.5: As in Figure 6.2, except for a lead time of $[45, 60]$ minutes. Base model for 5-km buffer is a random forest; base model for 0-km and 10-km buffers is a GBT ensemble.

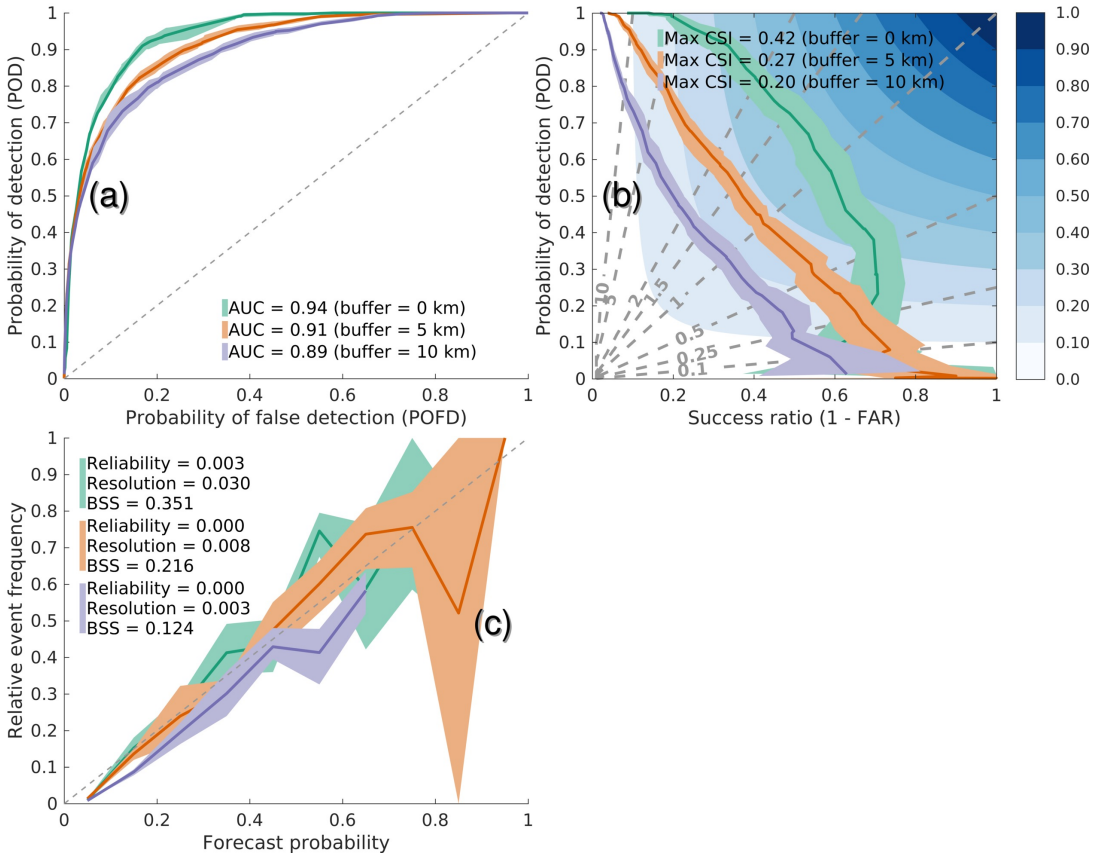

Figure 6.6: As in Figure 6.2, except for a lead time of $[60, 90]$ minutes. Base model for 10-km buffer is a random forest; base model for 0-km and 5-km buffers is a GBT ensemble.

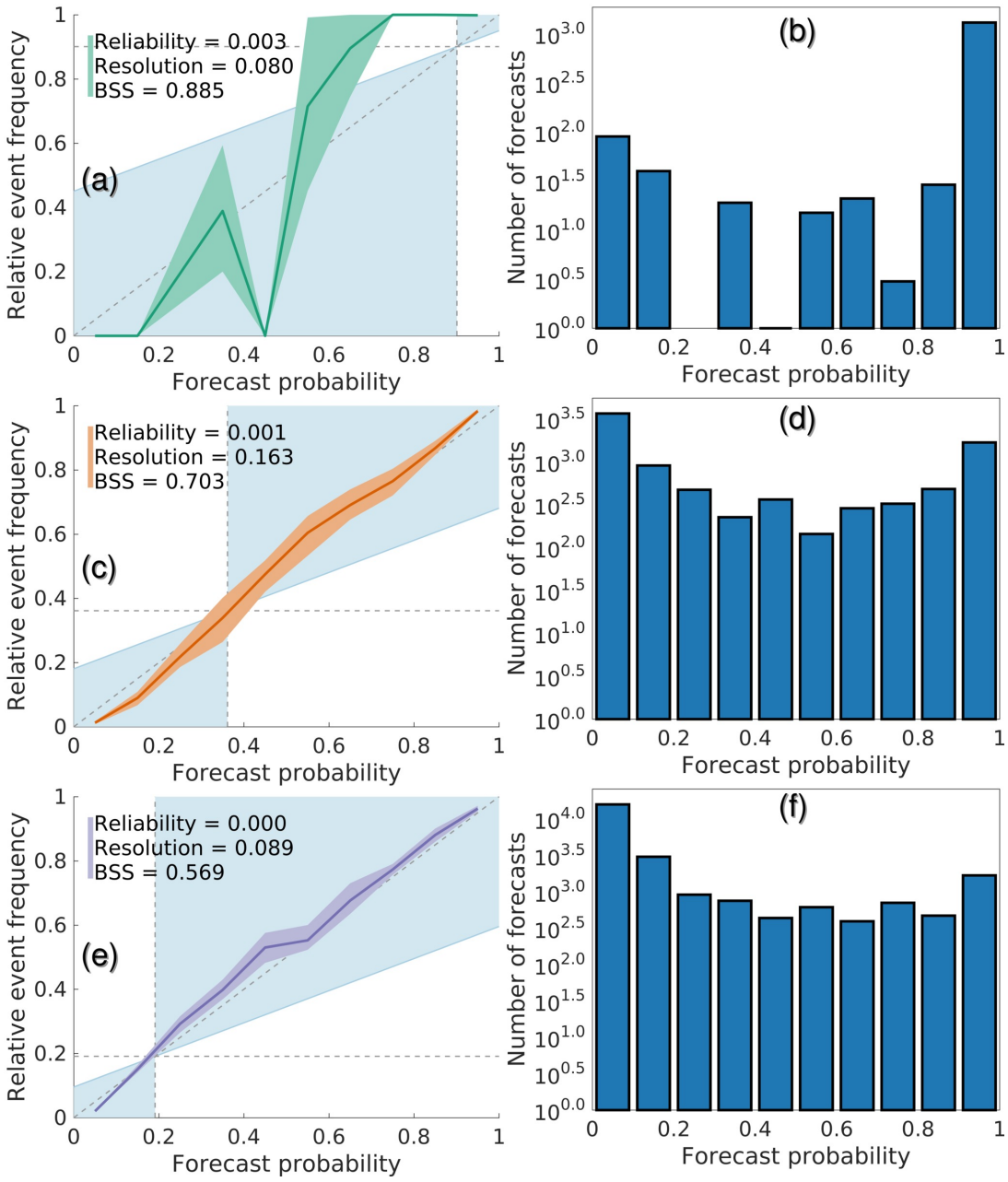

Figure 6.7: (a) Attributes diagram with (b) forecast histogram for buffer distance of 0 km; (c) and (d) same for 5-km buffer distance; (e) and (f) same for 10-km buffer distance. Lead-time window is $[0, 15]$ minutes in all cases. Each solid curve is a mean, and each shaded area is a 95% confidence interval, over 25 bootstrap replicates of the testing data. Base model for 0-km buffer is a random forest; base model for 5-km and 10-km buffers is a GBT ensemble.

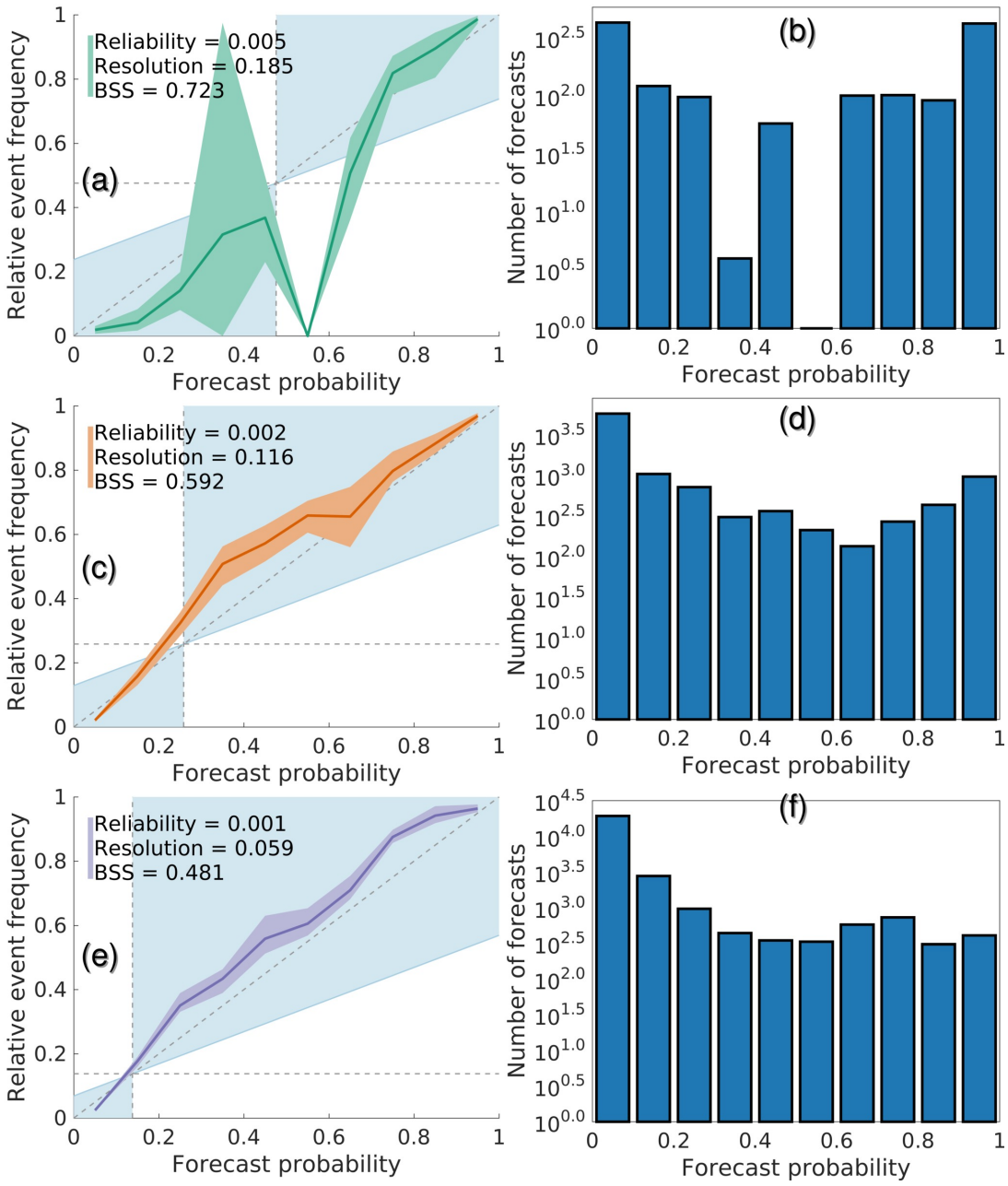

Figure 6.8: As in Figure 6.7, except for a lead time of $[15, 30]$ minutes. Base model for 0-km and 5-km buffers is a GBT ensemble; base model for 10-km buffer is LREN.

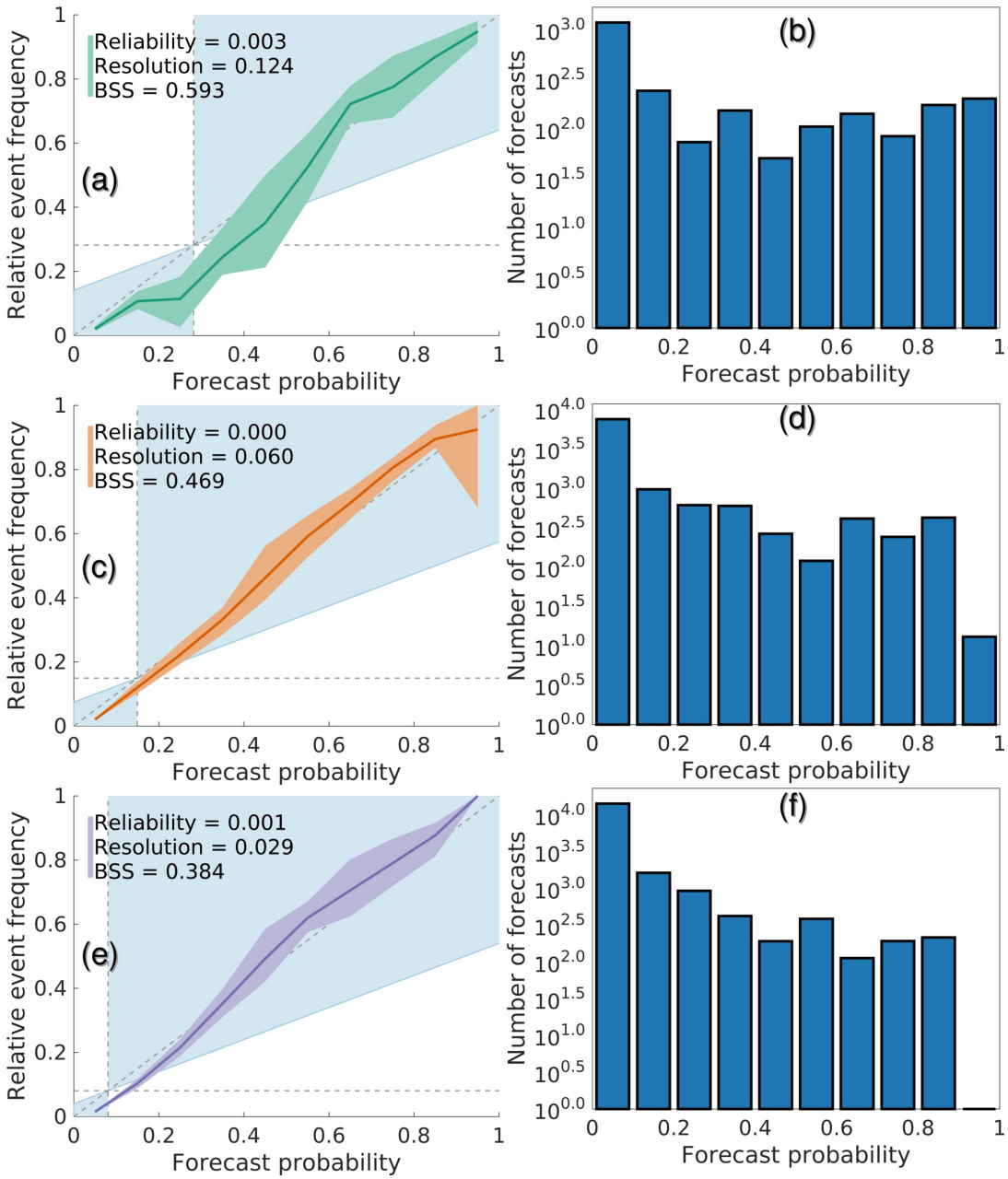

Figure 6.9: As in Figure 6.7, except for a lead time of $[30, 45]$ minutes. Base model for 0-km buffer is a random forest; base model for 5-km and 10-km buffers is an FFNN.

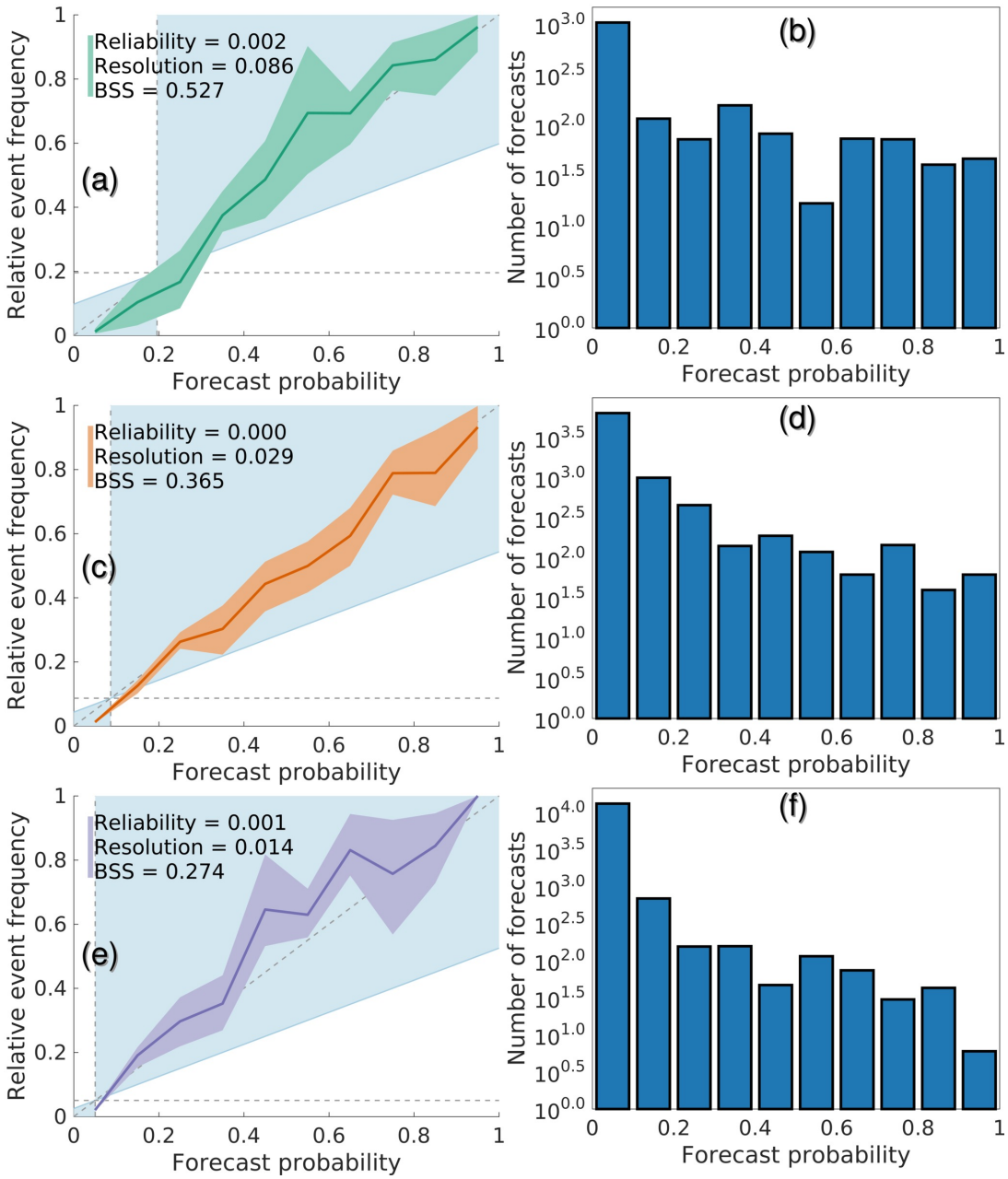

Figure 6.10: As in Figure 6.7, except for a lead time of $[45, 60]$ minutes. Base model for 5-km buffer is a random forest; base model for 0-km and 10-km buffers is a GBT ensemble.

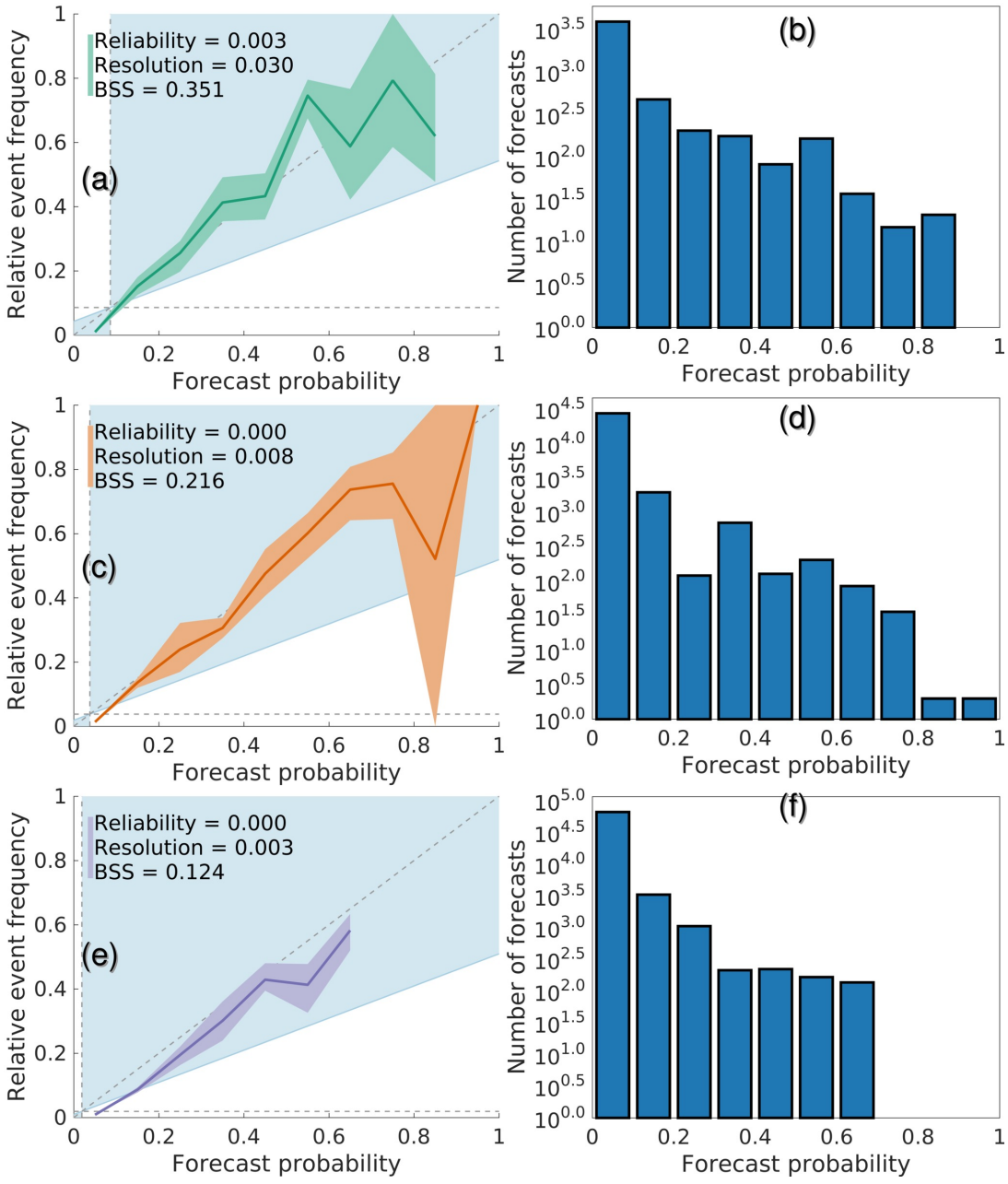

Figure 6.11: As in Figure 6.7, except for a lead time of $[60, 90]$ minutes. Base model for 10-km buffer is a random forest; base model for 0-km and 5-km buffers is a GBT ensemble.

Parameter	Value
Base model	Logistic regression
Calibration model	Isotonic regression
Latency time	5 selection steps
Verification statistic	AUC

Table 6.3: Parameters for sequential forward selection.

6.2.1 Sequential Forward Selection

SFS is run with the parameters listed in Table 6.3. To understand the role of the latency time and verification statistic, see Algorithm 5.1. The verification statistic is AUC for reasons discussed in Section 6.1.1. Sampling techniques are as described in Section 6.1. To reduce computing time, cross-validation is done only for the isotonic model, using the two-fold method discussed in Section 6.1.2. Of the two cross-validated models, the one with the lowest Brier score is kept, for reasons discussed in Section 6.1.1.

For each buffer distance and lead-time window, the procedure shown in Figure 6.12 is iterated 25 times. For each iteration, data from the uniform and best-observed distributions are bootstrapped independently. After 25 iterations, we count how many times each predictor variable was included in the model (*i.e.*, how many times it was included in `bestPredictors` by Algorithm 5.1).

6.2.2 Decision-tree Method

The vast majority of models selected in Experiment 1 are random forests and GBT ensembles (Section 6.1.3), which are based on decision trees and therefore eligible for the DTM. First we considered the following procedure for each buffer distance and lead-time window.

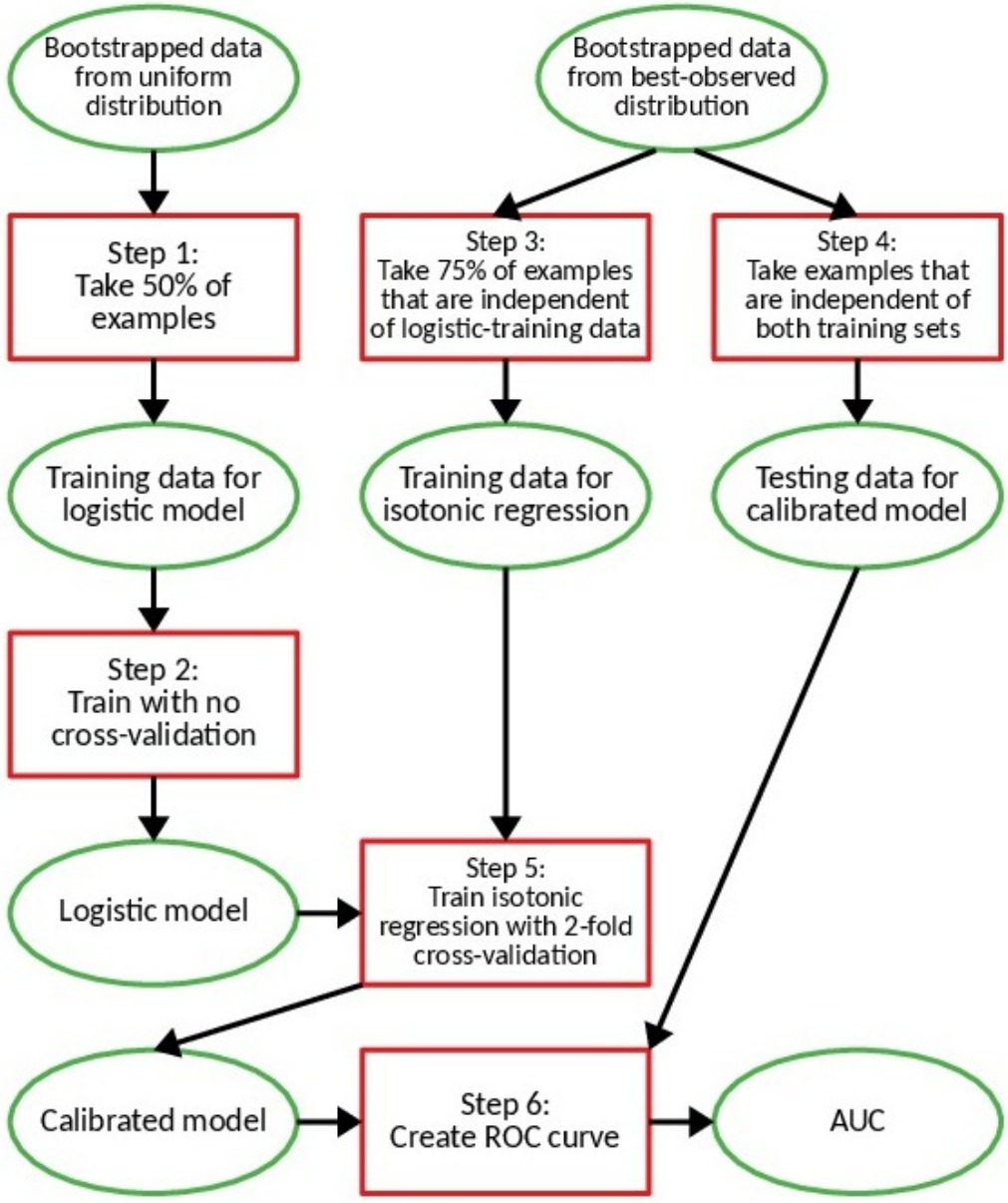

Figure 6.12: Flow chart for Experiment 2. Each red box is an action, and each green ellipse is an object (or set of objects). Bootstrapping and steps 1-6 are repeated 25 times for each buffer distance and lead-time window.

1. Find the best model (F_1) from Experiment 1.
2. Bootstrap the original training data[1] K times. Apply the DTM to each bootstrap replicate, using F_1 as the underlying model.

However, this would be inappropriate, because random forests already involve bootstrapping (called "tree-bagging" in Section 4.1.1.5). Thus, we use multiple models to obtain confidence intervals for variable importance. Specifically, for each buffer distance and lead-time window:

1. Find all random forests and GBT ensembles within 0.01 of the maximum AUC (Section 6.1.1). Let this set of models be F.
2. Run the DTM for each model in F.

Since the DTM applies only to decision trees, no isotonic regression or other probability calibration is used.

6.2.3 *J*-measures

Data are sampled from the best-observed distribution (Section 4.2.2), with $N^*_{obs} = 25$ as for Experiment 1 (Section 6.1). For each buffer distance and lead-time window, J-measures are calculated for 25 bootstrap replicates from this distribution. Each J-measure is calculated with 10 bins ($K = 10$ in Equation 5.3) and an equal number of storm objects per bin.

6.2.4 From Variable-ranking to Selection

Each method described above returns a set of variable importances: number of times included in model (SFS), mean reduction in deviance (DTM), and mean J-measure. For reasons discussed in Section 5.3, the linear-dependence-controlling (LDC) method is used to select variables with the highest J-measures, while

[1]The decision-tree method requires the original training data.

the linear-dependence-agnostic (LDA) method is used for all sets of variable importances.

At first we used the LDA method with the third criterion in Section 5.3.1, which stops at k variables when there is a large decrease in importance between the k^{th} and $(k+1)^{\text{th}}$-ranked variables. However, this often led to too few (1-3) or too many (> 20) variables being selected, which made results difficult to interpret. Thus, for each set of variable importances, we manually find a large decrease near 10 variables and select all variables above this cutoff.

6.2.5 Results

Figure 6.13 shows the four sets of variables selected for a buffer distance of 0 km and lead time of $[0, 15]$ minutes; Figure 6.14 shows results for 5 km and $[30, 45]$ minutes; and Figure 6.15 shows results for 10 km and $[60, 90]$ minutes. These are the smallest, median, and largest values respectively of both buffer distance and lead-time window. Thus, progressing from Figure 6.13 to 6.15, results are valid for increasingly distant predictions (both spatially and temporally). Results for other combinations of buffer distance and lead time are shown in Appendix C.

Observations from Figure 6.13 (shortest buffer distance and lead time) are as follows.

1. For J-measures with the linear-dependence-agnostic (LDA) selection, all variables selected are radar statistics. Seven of the nine statistics involve either -10 °C or -20 °C reflectivity, and the other two involve either maximum estimated hail size (MESH) or severe-hail index (SHI). None of these statistics involve gradients or a moment higher than the 0^{th} (mean).

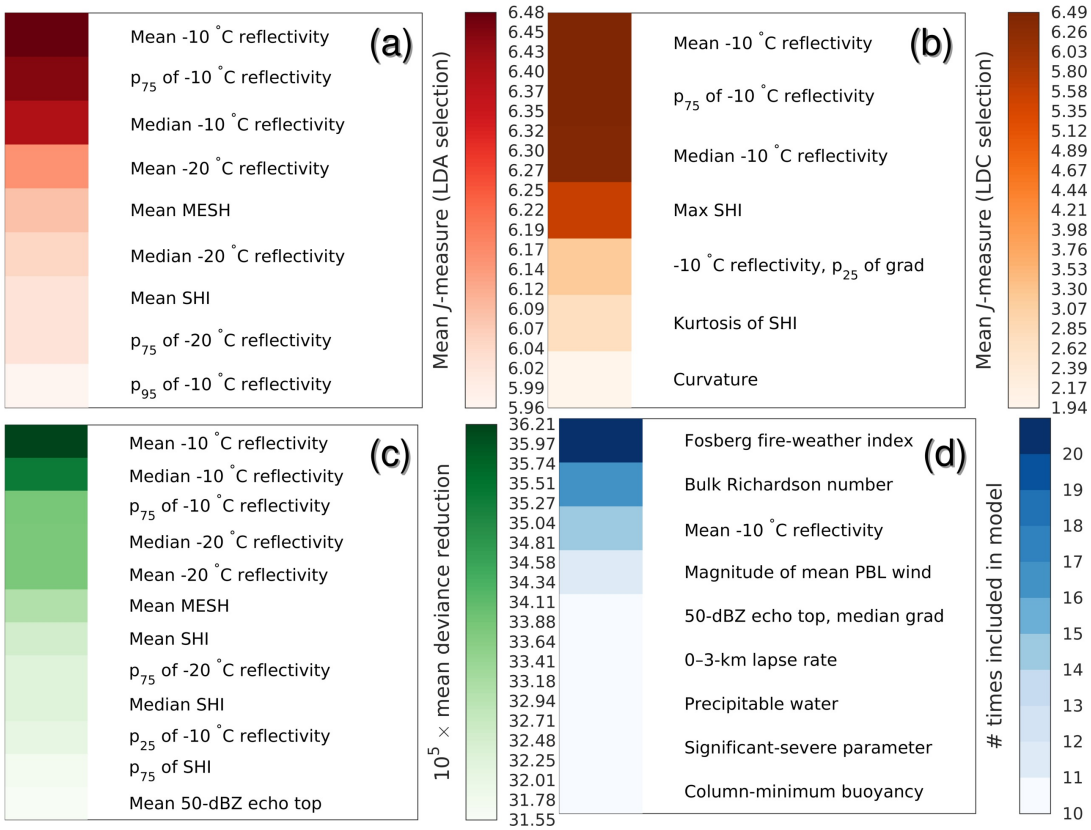

Figure 6.13: Variables selected for (a) J-measures with LDA selection; (b) J-measures with LDC selection; (c) decision-tree method; and (d) sequential forward selection at buffer distance of 0 km and lead time of $[0, 15]$ minutes.

2. For J-measures with the linear-dependence-controlling (LDC) selection method, six of the seven variables are radar statistics. Three of these statistics involve absolute values, and one involves gradients, of -10 °C reflectivity. The other two statistics involve absolute values of SHI (maximum and kurtosis). The last variable selected is the mean absolute curvature of vertices in the storm object (Section 3.5).
3. For the decision-tree method (DTM), all 12 variables selected are radar statistics. Seven of these involve either -10 °C or -20 °C reflectivity; four involve MESH or SHI; and the last involves 50-dBZ echo tops. Again, none of these statistics involve gradients or a moment higher than the 0^{th}.
4. Sequential forward selection (SFS) has very different results than the other methods. Of the nine variables selected, only two (mean -10 °C reflectivity and median gradient of 50-dBZ echo tops) are radar statistics; the other seven are sounding parameters. Three of the sounding parameters (0–3-km lapse rate, precipitable water, and column-minimum buoyancy) are thermodynamic, involving only vertical profiles of temperature and humidity; one (magnitude of mean planetary boundary layer [PBL] wind) is dynamic, involving only the vertical profile of wind velocity; and three (Fosberg fire-weather index, bulk Richardson number, and significant-severe parameter) are composite indices, involving both dynamic and thermodynamic variables.

Observations from Figure 6.14 (median buffer distance and lead time) are as follows.

1. For J-measures with LDA selection, results are nearly the same as in Figure 6.13. All variables selected are radar statistics; most involve -10 °C or -20 °C reflectivity, while the others involve MESH or SHI; none involve gradients; and none involve a moment higher than the 0^{th}.

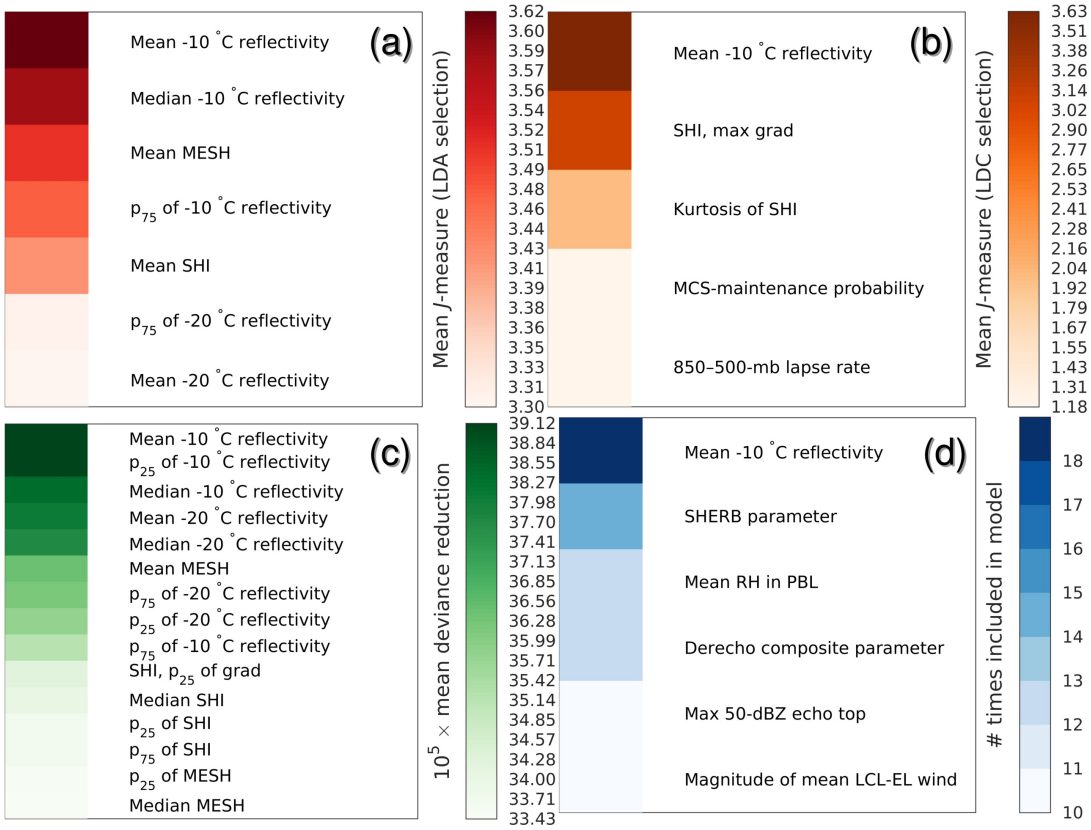

Figure 6.14: As in Figure 6.13, except for a buffer distance of 5 km and lead time of $[30, 45]$ minutes.

2. For J-measures with LDC selection, the top three of five variables are radar statistics. As in Figure 6.13, these statistics involve either -10 °C reflectivity or SHI. The other two variables (mesoscale convective system [MCS] maintenance probability and 850–500-mb lapse rate) are sounding parameters, which is different than in Figure 6.13.
3. For the DTM, results are nearly the same as in Figure 6.13. All variables selected are radar statistics; most involve -10 °C reflectivity, -20 °C reflectivity, MESH, or SHI; almost none involve gradients; and none involve a moment higher than the 0^{th}.
4. Again, SFS has very different results than the other methods. Of the six variables selected, only two (mean -10 °C reflectivity and maximum 50-dBZ echo top) are radar statistics; the other four are sounding parameters. One of the sounding parameters (mean PBL relative humidity) is purely thermodynamic; one (magnitude of mean wind from the lifting condensation level – equilibrium level [LCL-EL]) is purely dynamic; and the other two (severe hazards in environments with reduced buoyancy [SHERB] parameter and derecho composite parameter) are composite indices.

   The only variable selected in both Figures 6.13d (SFS at shortest buffer distance and lead time) and 6.14d (SFS at median buffer distance and lead time) is mean -10 °C reflectivity. However, a statistic involving 50-dBZ echo top is also selected in both figures (median gradient in 6.13d and maximum absolute value in 6.14d). The following variables are selected in Figure 6.13d with no similar counterpart in 6.14d: Fosberg fire-weather index, bulk Richardson number, magnitude of mean PBL wind, 0–3-km lapse rate, precipitable water, significant-severe parameter, and column-minimum buoyancy. Meanwhile, the following variables are selected in Figure 6.14d with no similar counterpart in 6.13d: SHERB parameter,

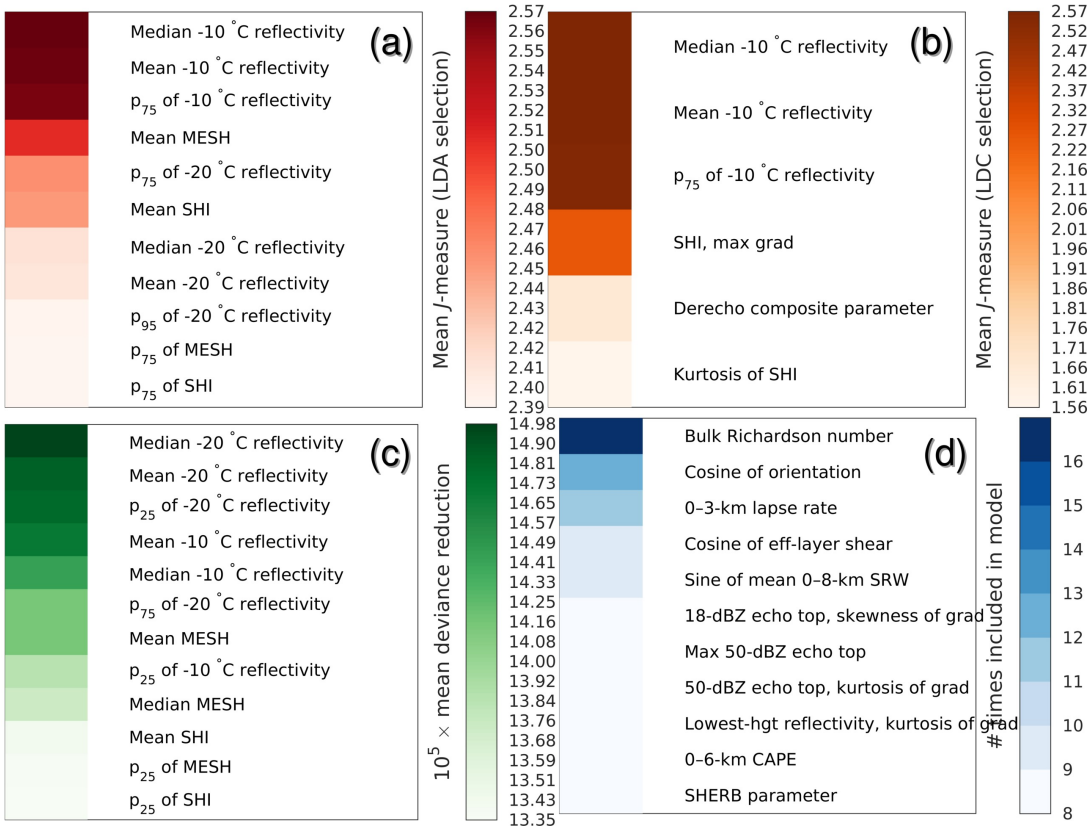

Figure 6.15: As in Figure 6.13, except for a buffer distance of 10 km and lead time of $[60, 90]$ minutes.

mean PBL relative humidity, derecho composite parameter, and magnitude of mean LCL-EL wind.

Observations from Figure 6.14 (longest buffer distance and lead time) are as follows.

1. For J-measures with LDA selection, results are basically the same as in Figures 6.13 and 6.14.
2. For J-measures with LDC selection, five of the top six variables are statistics involving -10 °C reflectivity and SHI, similar to Figures 6.13 and 6.14. The remaining variable is the derecho composite parameter.
3. For the DTM, results are basically the same as in Figures 6.13 and 6.14.

4. SFS produces very different results than the other methods. Of the 11 variables selected, only four (skewness of gradient of 18-dBZ echo top, maximum 50-dBZ echo top, kurtosis of gradient of 50-dBZ echo top, and kurtosis of gradient of lowest-altitude reflectivity) are radar statistics; one is a shape parameter (cosine of orientation); and the other six are sounding parameters. Two sounding parameters (0–3-km lapse rate and 0–6-km convective available potential energy [CAPE]) are purely thermodynamic; two (cosine of effective-layer shear and sine of mean 0–8-km storm-relative wind [SRW]) are purely dynamic; and two (bulk Richardson number and SHERB parameter) are composite indices.

   No variable is selected by SFS in all three figures. However, at least one statistic involving 50-dBZ echo top is selected in all three figures (median gradient in 6.13d; maximum absolute value in 6.14d and 6.15d; kurtosis of gradient in 6.15d). The following variables are selected in Figure 6.15d with no similar counterpart in 6.13d: cosine of orientation, cosine of effective-layer shear, sine of 0–8-km mean SRW, and 0–6-km CAPE.

To facilitate physical interpretation of the relationships found above, class-conditional PDFs for some of the selected variables are shown in Figures 6.16-6.18.

Figure 6.16 shows class-conditional PDFs of four statistics involving -10 °C reflectivity, which are selected very often for all buffer distances and lead times. PDFs in Figure 6.16 are conditioned on the storm-object label for a buffer distance of 5 km and lead time of $[30, 45]$ minutes, which are median values. Analogous PDFs for other buffer distances and lead times are similar: the non-severe PDF peaks at low reflectivity values, and the severe PDF peaks at high reflectivity values. This makes physical sense, because reflectivity is positively correlated

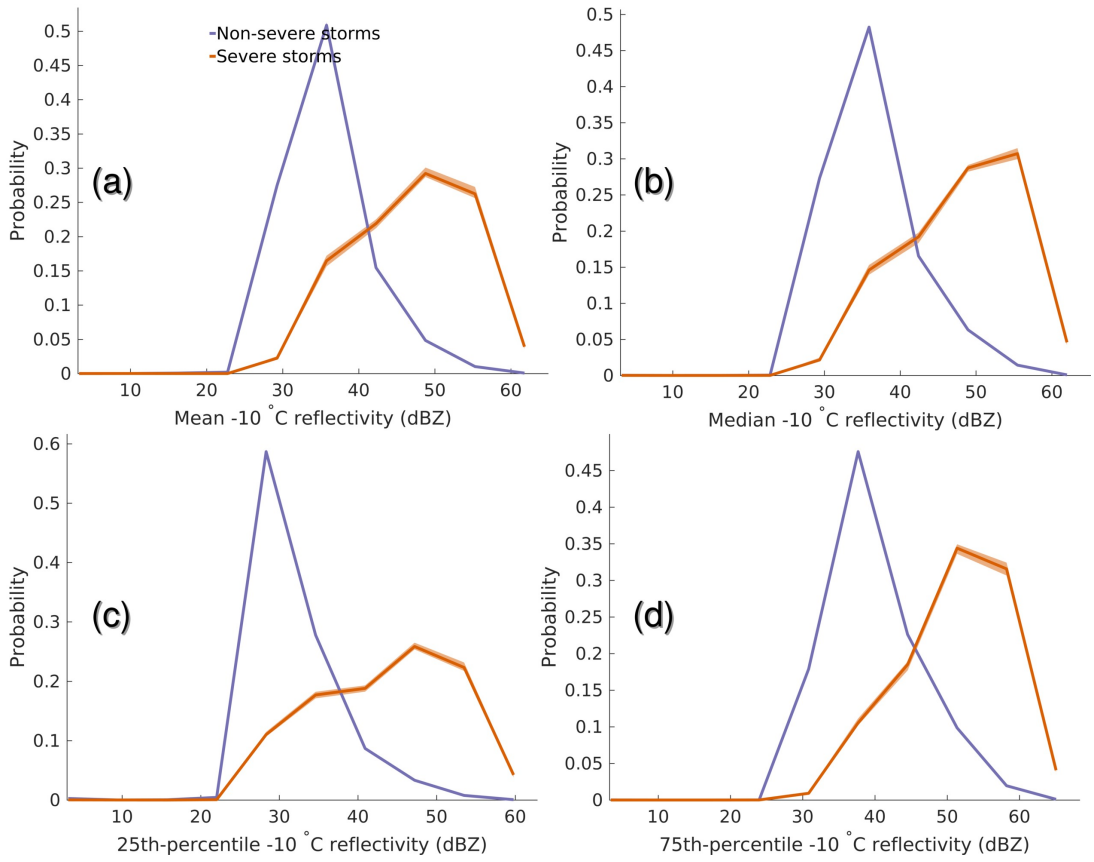

Figure 6.16: Class-conditional PDFs for statistics involving -10 °C reflectivity. The "class" is the label for the storm object (1 if $U_{max} \geq 50$ kt, 0 otherwise) at a buffer distance of 5 km and lead time of $[30, 45]$ minutes. The orange curve is the PDF for storm objects with label = 1; the purple curve is for storm objects with label = 0. Each solid line is a mean, and each shaded area is a 95% confidence interval, over 25 bootstrap replicates.

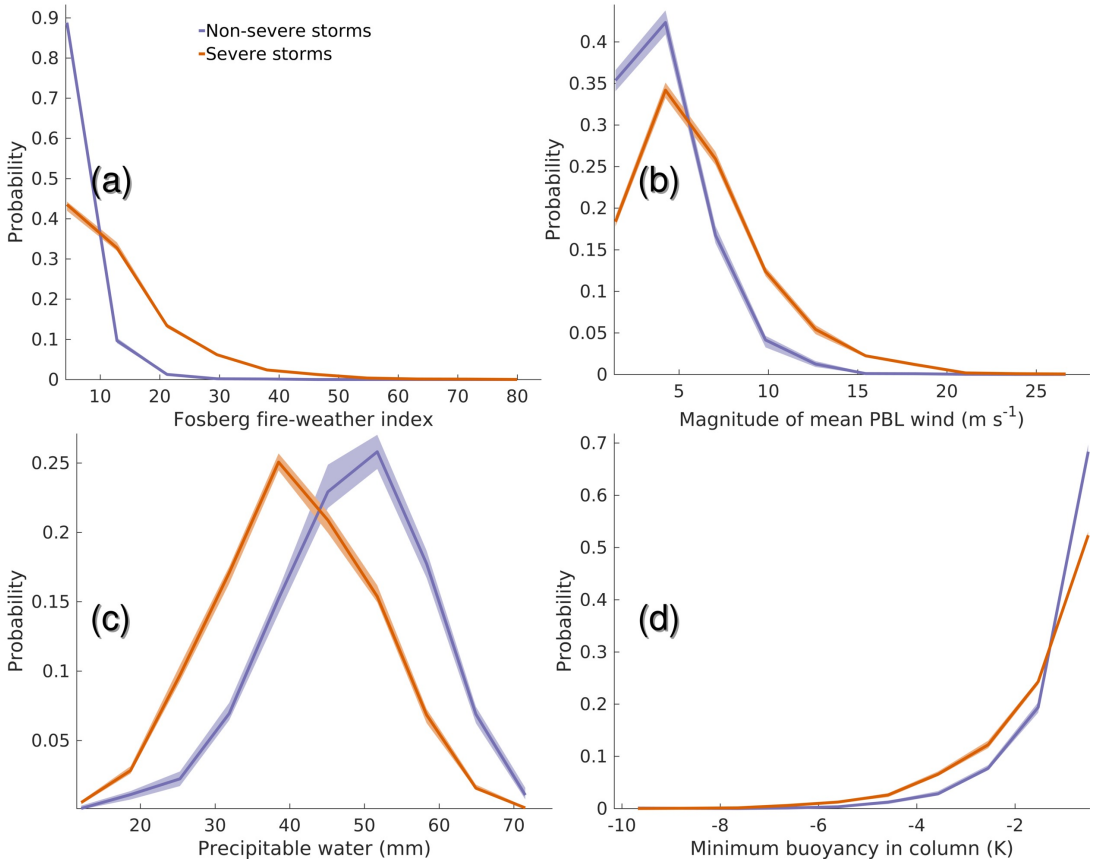

Figure 6.17: Class-conditional PDFs of variables selected for buffer distance of 0 km and lead time of $[0, 15]$ minutes. Meanings of solid lines and shaded areas are as in Figure 6.16.

with updraft strength (Section 3.1.1), which is positively correlated with downdraft strength and the potential for strong horizontal wind caused by downbursts (Section 2.1).

Statistics involving -20 °C reflectivity, MESH, and SHI are also selected very often for all lead times and buffer distances. Class-conditional PDFs for these variables (Figures C.13-C.15) are similar to those for -10 °C reflectivity. Figures C.13-C.15 can be interpreted in essentially the same way as Figure 6.16, since all three variables are positively correlated with updraft strength.

Figure 6.17 shows class-conditional PDFs of variables selected mainly at the shorter buffer distances and lead times. Accordingly, these PDFs are conditioned on the storm-object label for 0 km and $[0, 15]$ minutes. This figure is discussed below.

1. The severe PDF peaks at higher values of the Fosberg fire-weather index (FFWI), and the non-severe PDF peaks at lower FFWI values. In other words, FFWI is positively correlated with severe wind. This makes sense, because FFWI decreases with surface relative humidity (RH) and increases with surface wind speed[2]. When surface RH is low, there is usually ample subsaturated air below the cloud base, which increases the potential for evaporative cooling and damaging downbursts (Section 2.1). Also, when the background wind speed is high, less storm-related enhancement is needed to reach a wind speed of 50 kt.
2. Severe wind is positively correlated with mean wind in the planetary boundary layer (PBL). The explanation for this relationship is the same as for surface wind speed in point 1.
3. Severe wind is negatively correlated with precipitable water (PW), which is the amount of water vapour per unit horizontal area. (Assuming a liquid-water density of 1000 kg m^{-3}, 1 mm of PW = 1 kg m^{-2} of PW, which makes the "per unit horizontal area" clearer.) When the PW is higher, low-level humidity tends to be higher, which means that there is less subsaturated air below the cloud base – therefore, less potential for evaporative cooling and damaging downbursts.

[2]Background wind speed, not associated with the storm itself. The exact calculation method for FFWI is shown in the SHARPpy documentation (`https://github.com/sharppy/SHARPpy/blob/master/sharppy/sharptab/fire.py`).

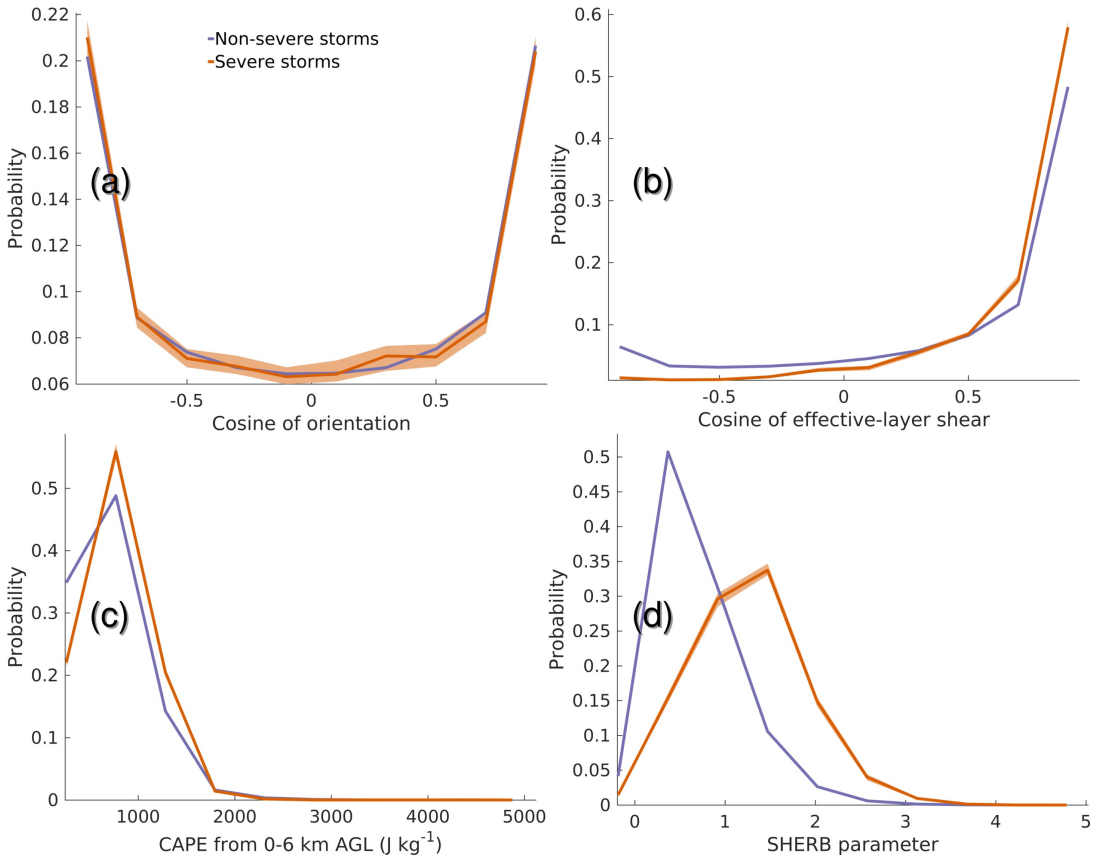

Figure 6.18: Class-conditional PDFs of selected variables for buffer distance of 10 km and lead time of $[60, 90]$ minutes. Meanings of solid lines and shaded areas are as in Figure 6.16.

4. Severe wind is positively correlated with column-minimum buoyancy. When the column-minimum buoyancy is higher (less negative), buoyancy throughout the column (not just at the level of minimum buoyancy) tends to be higher, which means that there is less negative buoyancy – therefore, less resistance to convection and more potential for strong thunderstorms.

Figure 6.18 shows class-conditional PDFs of variables selected mainly at the longer buffer distances and lead times. Accordingly, these PDFs are conditioned on the storm-object label for 10 km and $[60, 90]$ minutes. In the following discussion, keep in mind that the cosine of a vector is proportional to its x-component

(east-west) and the sine of a vector is proportional to its y-component (north-south). East is defined as the positive x-direction, and north is defined as the positive y-direction.

$$u_x = \|\vec{u}\|\cos(\vec{u}) \tag{6.1a}$$

$$u_y = \|\vec{u}\|\sin(\vec{u}) \tag{6.1b}$$

Figure 6.18 is discussed below.

1. Cosine of orientation ($\cos(\theta_{orient})$). This is the cosine between the x-axis and the storm's major axis. When $|\cos(\theta_{orient})| < \frac{\sqrt{2}}{2}$, $\theta_{orient} \in (45, 135)^\circ$, which means that the storm is oriented mainly along the y-axis (north-south). When $|\cos(\theta_{orient})| > \frac{\sqrt{2}}{2}$, $\theta_{orient} < 45^\circ$ or $> 135^\circ$, so the storm is oriented mainly along the x-axis (east-west). However, the relationship between θ_{orient} and severe wind is difficult to interpret, because the severe and non-severe PDFs are essentially the same. Despite this, cosine of orientation was included in 12 of 25 models by sequential forward selection at a buffer distance of 10 km and lead time of $[60, 90]$ minutes (Figure 6.15). Perhaps θ_{orient} is important for only a small subset of storms, which is why there is no clear correlation in Figure 6.18a.
2. Severe wind is positively correlated with the cosine (x-component) of eff-layer shear. In general, wind shear is important for storm longevity, because it allows the updraft and downdraft columns to separate, which prevents the downdraft from extinguishing the updraft. Also, when thunderstorms occur in a high-shear environment in the CONUS, the shear vector is often westerly, due to southerly winds near the surface and westerly winds aloft. A westerly shear vector points eastward, which means that it has a high cosine. Thus, it makes physical sense that the cosine of effective-layer shear is positively correlated with severe wind.

3. Severe wind is positively correlated with 0–6-km CAPE. Higher CAPE means more positive buoyancy, which means more potential for strong thunderstorms.
4. Severe wind is positively correlated with the SHERB parameter. The SHERB parameter increases with 0–3-km lapse rate, 700–500-mb lapse rate, and effective-bulk-layer shear[3]. When low-level lapse rate is high (close to the dry adiabatic lapse rate), there tends to be ample subsaturated air below the cloud base, which increases the potential for evaporative cooling and downbursts. Also, as mentioned above, high shear is required for storm maintenance.

Finally, we have three general observations from Experiment 2. First, two of four variables selected primarily for longer lead times (cosine of effective-layer shear and the SHERB parameter, as shown in Figure 6.18) are strongly related to storm longevity. This makes physical sense, because to predict storm-scale winds at a lead time of $[60, 90]$ minutes, a significant subproblem is predicting whether or not the storm will last long enough.

Second, statistics involving -10 °C and -20 °C reflectivity are selected much more often than those involving any other radar variable. Also, statistics involving MESH and SHI are selected quite often. There are two possible explanations for this.

1. Hail growth is strongest around the -10 °C level (Jewell and Brimelow, 2009), so hail is most common in this area. Since reflectivity is the sixth moment of the drop-size distribution (Equation 3.1), it is dominated by large particles such as hail stones. (For example, a 10-mm-diameter hail

[3]The exact calculation method for the SHERB parameter is shown in the SHARPpy documentation (`https://github.com/sharppy/SHARPpy/blob/master/sharppy/sharptab/params.py`).

stone produces a reflectivity 10^6 times higher than a 1-mm-diameter rain drop.) Thus, -10 °C (and to a lesser extent, -20 °C) reflectivity are strongly correlated with hail size. When there is large hail, there is high potential for downbursts caused by evaporative cooling and especially precipitation drag (Section 2.1). This would also explain why statistics involving MESH and SHI are selected quite often.

2. The popularity of -10 °C and -20 °C reflectivity may be an artifact of storm detection, which is based on the -10 °C reflectivity field (Section 3.3). `w2segmotionll` creates the storm object by outlining a locally maximum area of -10 °C reflectivity, which probably contains the most salient features of -10 °C reflectivity. However, if the storm is tilted away from the vertical axis, this 2-D outline will not contain the most salient information at other height or temperature levels.

Third, two of four predictor types (radar statistics and sounding parameters) are selected quite often (several times for each buffer distance and lead-time window). However, shape parameters are rarely selected, and storm motion is never selected. There are two possible explanations.

1. There are many fewer shape parameters (9) and storm-motion components (3) than radar statistics (264) and sounding parameters (155).
2. The unpopularity of shape parameters may be an artifact of storm detection. As mentioned above, each storm object is a 2-D outline based only on the -10 °C reflectivity field, which may not be a good representation of storm shape at other height and temperature levels.

6.3 Experiment 3: Spring 2016 Forecasting Experiment

Our models were demonstrated in the National Oceanic and Atmospheric Administration's (NOAA) 2016 Spring Forecasting Experiment (SFE). The SFE is an annual event in which human forecasters test various NWP models, machine-learning models, and other forecasting methods for thunderstorm-related hazards (Clark et al., 2012).

There are four differences between the archived data used for model development (Table 3.1) and real-time data used for the SFE. First, real-time radar images are from the Multi-radar Multi-sensor (MRMS) (Smith et al., 2016) dataset, rather than MYRORSS. The main differences between MRMS and MYRORSS are listed below.

1. MRMS undergoes slightly less quality control, because it must be available in near-real time.
2. MRMS has a spatial resolution of 0.005° (~0.5 km) for low-level and mid-level azimuthal shear, whereas MYRORSS has a resolution of 0.01° (~1.0 km). The spatial resolution for all other variable is 0.01° (~1.0 km) in both MRMS and MYRORSS.
3. MRMS has a temporal resolution of ~2 minutes, compared to 5 minutes for MYRORSS.

Second, near-storm environment (NSE) soundings are interpolated from the Rapid Refresh (RAP) model, rather than the Rapid Update Cycle (RUC). The RUC was discontinued in April 2012, and the RAP is its successor. RAP soundings are temporally interpolated between forecast times, rather than initialization times. In other words, if the latest RAP solution was initialized at 1200

UTC and soundings are needed for 1330 UTC, they are temporally interpolated from the 1- and 2-hour forecasts (valid at 1300 and 1400 UTC, respectively). During model development (Section 3.5), soundings would have been interpolated from the 0-hour RUC analyses initialized at 1300 and 1400 UTC. Clearly this is not an option in real-time, since the 1400 UTC initialization is not available at 1330 UTC. Also, the NARR could not be used as a backup model in real-time, because it (like all other reanalyses) does not operate in real-time.

Third, wind observations are not linked to storm cells. During model development, the point of linking wind observations is to create labels, which are used to train the machine-learning models. However, in real-time the models are deployed, rather than trained. In other words, models are used to predict unknown labels, rather than being trained to learn the relationship between predictors and labels. Also, labeling would be impossible in real-time, because labels are based on future wind observations (at lead times of $[0, 15]$; $[15, 30]$; $[30, 45]$; $[45, 60]$; and $[60, 90]$ minutes).

Fourth, storm tracks are based only on `w2segmotionll`, not `w2besttrack`. Since `w2besttrack` is a post-event algorithm, it requires information from both the past and future, which is clearly not available in real-time. In any case, storm-tracking is unimportant in real-time, because its main purpose is to create labels for model development. The only purpose of tracking in real-time is to find the motion vector for each storm cell, which is used as a predictor (Section 3.5). For each storm object S_k, motion is estimated by a first-order backward difference ($\frac{\text{position of } S_k - \text{position of } S_{k-1}}{\text{time of } S_k - \text{time of } S_{k-1}}$), where S_{k-1} is the previous object in the same track. If S_k is the first object in the track, storm motion is left blank and not used as a predictor.

To select models for the 2016 SFE, we performed an experiment similar to Experiment 1. For each buffer distance and lead-time window, we trained several

base models with different sets of parameters, calibrated each base model using isotonic regression, and selected the combined model (base model + isotonic regression) with the highest AUC. The main difference is that in the pre-SFE experiment we tested only two base models (random forests and GBT ensembles); also, we tested fewer sets of parameters for these models. For this reason the best SFE models were slightly inferior to the best models from Experiment 1. For 0 km and $[0, 15]$ minutes, the best SFE model had an AUC of 0.975, maximum CSI of 0.91, and BSS of 0.74 (compared to 0.996, 0.99, and 0.88 respectively). For 10 km and $[60, 90]$ minutes, the best SFE model had an AUC of 0.87, maximum CSI of 0.18, and BSS of 0.09 (compared to 0.89, 0.20, and 0.12 respectively).

Forecast probabilities from the best SFE models were shown in the Probabilistic Hazard Information (PHI) (Karstens et al., 2014) tool, which is a visualization tool for storm data and forecasts. Specifically, a probability-time graph was shown for each storm object (*e.g.*, Figure 3.16). Forecasts were updated whenever new MRMS data arrived (every ~2 minutes), and the latency time (processing time for each forecast update) was ~4 minutes. Since the SFE ended in July, we have not had time to verify the SFE forecasts. This is a key priority in the near future.

Chapter 7

Conclusions

This study combined three datasets – radar images from MYRORSS, atmospheric soundings from the RUC and NARR, and near-surface wind observations from both weather stations and NWS local storm reports – to predict damaging straight-line winds from thunderstorms. These datasets were processed in four ways. First, storm cells were detected from the radar images and tracked in time. Second, wind observations were linked to nearby storm cells. Third, a set of 431 predictors (including radar statistics, storm motion, shape parameters describing the storm boundary, and sounding parameters describing the near-storm environment) was created for each storm object (one storm cell at one time step). Fourth, labels were calculated for each storm object S, indicating whether or not S was linked to a severe wind gust ($\geq$ 50 kt) at the given buffer distance and lead-time window. Since there were three buffer distances (0, 5, and 10 km) and five lead-time windows ($[0, 15]$; $([15, 30]$; $([30, 45]$; $([45, 60]$; and $([60, 90]$ minutes), 15 labels were calculated for each storm object.

These predictors and labels were used to train a combined model for each buffer distance and lead-time window. Each combined model had two components: a base model, which produced the original forecasts (probability of maximum storm-object wind $U_{max} \geq 50$ kt), and an isotonic-regression model,

which calibrated these forecasts into reliable probabilities. We experimented with five base models (logistic regression, logistic regression with an elastic net, feed-forward neural nets, random forests, and gradient-boosted tree [GBT] ensembles) and determined that, for most buffer distances and lead-time windows, the best base models are random forests and GBT ensembles.

Finally, we used three methods to select the most important predictors for each buffer distance and lead-time window: sequential forward selection (SFS), the decision-tree method (DTM), and J-measures. Two of the four predictor types (radar statistics and sounding parameters) were frequently selected as important variables. Shape parameters were very rarely selected, and storm motion was never selected. Statistics of -10 °C reflectivity, -20 °C reflectivity, maximum estimated hail size (MESH), and severe-hail index (SHI) – all of which are strongly related to hail – were selected more often than other radar statistics. However, this may be an artifact of storm detection, which is based only on the -10 °C reflectivity field.

Our hypothesis was that for each combination of buffer distance, lead-time window, and forecast probability, our models would outperform climatology. According to the Brier skill score (which is positive – better than climatology – in the blue shaded area of the attributes diagram), this is true for all but a few combinations.

To our knowledge, only four previous studies (Kitzmiller et al., 1995; Marzban and Stumpf, 1998; Alexiuk et al., 1999; Cintineo et al., 2014) have used machine learning to predict damaging straight-line convective winds. We cannot compare our performance directly to these models, because they either (a) forecast many thunderstorm hazards at once, rather than focusing specifically on straight-line wind (Kitzmiller et al., 1995; Marzban and Stumpf, 1998; Cintineo et al., 2014), or (b) forecast straight-line wind only for storms that were already known to be

severe (Alexiuk et al., 1999). However, we made clear contributions beyond these studies: (a) using measured wind observations from weather stations, rather than only human reports; (b) using multiple data sources to create predictors; (c) predicting only straight-line wind, separate from other thunderstorm hazards[1]; (d) using composited, rather than single-radar, data; (e) using advanced machine-learning algorithms with probability calibration; and (f) using a large number of predictors.

Planned work for the near future is listed below.

1. Interpolation of sounding data. Currently, to predict winds at a lead time of $[\Delta t_{min}, \Delta t_{max}]$ for a storm object at t_0, we interpolate sounding data to the storm object at t_0. Instead, we should extrapolate the storm along its motion vector to some time in the forecast window $[t_0 + \Delta t_{min}, t_0 + \Delta t_{max}]$, then interpolate sounding data to this new position and time.
2. Interpolate storm-cell-wise probabilities to a grid. This would allow forecasts to be composited over the three buffer distances, which would facilitate interpretation. Also, this would allow disjoint spatial buffers to be used (since forecaster interpretation of spatial buffers [Section 3.7] would no longer be an issue), which would allow the predictor-predictand relationships at different spatial scales to be properly separated.

[1]It would be interesting to use our methodology to predict other storm-related hazards (*e.g.*, tornadoes, hail, lightning, and aircraft turbulence). If the predictor-predictand relationships were the same across hazards, this would support the use of a single model for multiple severe-weather types. However, verification data for some of these hazards (especially hail, which relies on human reports, and aircraft turbulence, which relies on proprietary data from airlines) are very difficult to obtain. In any case, different physical mechanisms are responsible for each hazard, so our physical intuition is that "one size does not fit all" for predicting storm-related hazards.

3. Better verification data. Currently we are using NWS severe-wind reports from `http://www.spc.noaa.gov/climo/online`, which are of lower quality than those in the *Storm Data* publication.
4. Add variable-selection methods. We used only three methods (SFS, the DTM, and J-measures), mainly because we were limited by computing time. However, SFS results are very different than those of other methods, which suggests that more methods are needed to form a consensus.

In the less immediate future we may also make the following enhancements.

1. Storm detection and tracking. We did not experiment with algorithms other than `w2segmotionll` and `w2besttrack`, and we did not experiment extensively with input parameters for these algorithms. Thus, we could probably find a better detection and tracking configuration.
2. A major strength of this study is that it combined several data sources (radar images, model soundings, and near-surface wind observations from four datasets). However, we could probably improve results by using new data sources (*e.g.*, satellite and dual-polarization radar data) as predictors. Also, we could use recent wind observations as predictors, which would be similar to temporal auto-regression (if the storm cell has produced severe wind in the past, it is more likely to produce severe wind in the future).
3. Sampling of storm objects. The "best-observed" distribution was our attempt to deal with a paucity of near-surface wind observations and sample only the best-observed storms, defined as those with ≥ 25 wind observations or a wind gust ≥ 50 kt. However, we did not consider spatial and temporal characteristics of wind observations linked to the storm cell. For example, if all wind observations occur at the same time or location, the storm should not be considered well observed, because it is very likely that these observations missed the highest winds produced by the storm.

4. More advanced machine learning. We have a very large dataset with many high-level objects and spatiotemporal properties, which is ideal for learning with spatiotemporal relational probability trees (McGovern et al., 2008). Also, we have a lot of three-dimensional radar and NWP model data, which are ideal for learning with convolutional neural nets (Ciresan et al., 2011).
5. More detailed and relevant predictions. The current system forecasts only the probability of severe wind ($\geq$ 50 kt) for each storm cell. In addition to interpolating storm-cell-wise probabilities to a grid (planned for near-future work), it would be useful to predict multiple threshold exceedances (*e.g.*, 30 kt for aviation applications) and the real value of the maximum storm wind.

Bibliography

Adrianto, I., T. Trafalia, and V. Lakshmanan, 2009: Support vector machines for spatiotemporal tornado prediction. *International Journal of General Systems*, **38 (7)**, 759–776.

Aha, D., and R. Bankert, 1996: A Comparative Evaluation of Sequential Feature Selection Algorithms. *Learning from Data*, D. Fisher, and H.-J. Lenz, Eds., Springer Science & Business Media.

Alexiuk, M., N. Pizzi, and W. Pedrycz, 1999: Classification of volumetric storm cell patterns. *Canadian Conference on Electrical and Computer Engineering*, Edmonton, AB, Canada, Institute of Electrical and Electronics Engineers.

Aligo, E., B. Ferrier, J. Carley, E. Rogers, M. Pyle, S. Weiss, and I. Jirak, 2014: Modified microphysics for use in high-resolution NAM forecasts. *Conference on Severe Local Storms*, Madison, WI, American Meteorological Society.

Anagnostou, E., 2004: A convective/stratiform precipitation classification algorithm for volume scanning weather radar observations. *Meteorological Applications*, **11 (4)**, 291–300.

Baldwin, M., J. Kain, and S. Lakshmivarahan, 2005: Development of an automated classification procedure for rainfall systems. *Monthly Weather Review*, **133 (4)**, 844–862.

Bankert, R., 1994: Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. *Journal of Applied Meteorology*, **33 (8)**, 909–918.

Benjamin, S., and Coauthors, 2004: An hourly assimilation-forecast cycle: The RUC. *Monthly Weather Review*, **132 (2)**, 495–518.

Beucher, S., 1982: Watersheds of functions and picture segmentation. *International Conference on Acoustics, Speech, and Signal Processing*, Paris, France, Institute of Electrical and Electronics Engineers.

Beucher, S., and C. Lantuejoul, 1979: Use of watersheds in contour detection. *International Workshop on Image Processing*, Rennes, France, Research Institute of Computer Science and Random Systems.

Billet, J., M. DeLisi, B. Smith, and C. Gates, 1997: Use of regression techniques to predict hail size and the probability of large hail. *Weather and Forecasting*, **12 (1)**, 154–164.

Bluestein, H., 1992: *Synoptic-Dynamic Meteorology in Midlatitudes*, Vol. 1. Oxford University Press.

Bowie, E., and R. Weightman, 1914: *Types of Storms of the United States and Their Average Movements*. U.S. Government Printing Office.

Breiman, L., 2001: Random Forests. *Machine Learning*, **45 (1)**, 5–32.

Bunkers, M., B. Klimowski, J. Zeitler, R. Thompson, and M. Weisman, 2000: Predicting Supercell Motion Using a New Hodograph Technique. *Weather and Forecasting*, **15 (1)**, 61–79.

Campbell, S., and S. Olson, 1987: Recognizing low-altitude wind shear hazards from Doppler weather radar: An artificial intelligence approach. *Journal of Atmospheric and Oceanic Technology*, **4 (1)**, 5–18.

Chisholm, D., J. Ball, K. Veigas, and P. Luty, 1968: The diagnosis of upper-level humidity. *Journal of Applied Meteorology*, **7 (4)**, 613–619.

Cintineo, J., M. Pavolonis, J. Sieglaff, and D. Lindsey, 2014: An empirical model for assessing the severe weather potential of developing convection. *Weather and Forecasting*, **29 (3)**, 639–653.

Ciresan, D., U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber, 2011: Flexible, high performance convolutional neural networks for image classification. *International Joint Conference on Artificial Intelligence*, Barcelona, Spain, International Joint Conference on Artificial Intelligence.

Clark, A., A. MacKenzie, A. McGovern, V. Lakshmanan, and R. Brown, 2015: An Automated, Multiparameter Dryline Identification Algorithm. *Weather and Forecasting*, **30 (6)**, 1781–1794.

Clark, A., and Coauthors, 2012: An overview of the 2010 Hazardous Weather Testbed Experimental Forecast Program Spring Experiment. *Bulletin of the American Meteorological Society*, **93 (1)**, 55–74.

Coniglio, M., D. Stensrud, and M. Richman, 2004: An observational study of derecho-producing convective systems. *Weather and Forecasting*, **19 (2)**, 320–337.

Corfidi, S., M. Coniglio, A. Cohen, and C. Mead, 2016: A proposed revision to the definition of "derecho". *Bulletin of the American Meteorological Society*, **97 (6)**, 935–949.

Crum, T., and R. Alberty, 1993: The WSR-88D and the WSR-88D operational support facility. *Bulletin of the American Meteorological Society*, **74 (9)**, 1669–1687.

Delanoy, R., and S. Troxel, 1993: Machine intelligent gust front detection. *Lincoln Laboratory Journal*, **6 (1)**, 187–212.

Efron, B., 1979: Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, **7 (1)**, 1–26.

Ferrero, E., L. Mortarini, M. Manfrin, M. Solari, and R. Forza, 2014: Physical simulation of atmospheric microbursts. *Journal of Geophysical Research: Atmospheres*, **119 (11)**, 6292–6305.

Freund, Y., and R. Schapire, 1997: A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55 (1)**, 119–139.

Friedman, J., 2001: Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29 (5)**, 1189–1232.

Fujita, T., 1990: Downbursts: meteorological features and wind field characteristics. *Journal of Wind Engineering and Industrial Aerodynamics*, **36 (1)**, 75–86.

Gagne, D., A. McGovern, J. Basara, and R. Brown, 2012: Tornadic supercell environments analyzed using surface and reanalysis data: a spatiotemporal relational data-mining approach. *Journal of Applied Meteorology and Climatology*, **51 (12)**, 2203–2217.

Gagne, D., A. McGovern, and J. Brotzge, 2009: Classification of convective areas using decision trees. *Journal of Atmospheric and Oceanic Technology*, **26 (7)**, 1341–1353.

Gagne, D., A. McGovern, J. Brotzge, M. Coniglio, J. C. Jr., and M. Xue, 2015: Day-ahead hail prediction integrating machine learning with storm-scale numerical weather models. *Conference on Artificial Intelligence*, Austin, Texas, Association for the Advancement of Artificial Intelligence.

Gagne, D., A. McGovern, J. Brotzge, and M. Xue, 2013: Severe hail prediction within a spatiotemporal relational data mining framework. *International Conference on Data Mining*, Dallas, TX, Institute of Electrical and Electronics Engineers.

Halbert, K., W. Blumberg, and P. Marsh, 2015: SHARPpy: Fueling the Python Cult. *5th Symposium on Advances in Modeling and Analysis Using Python*, Phoenix, AZ, American Meteorological Society.

Haykin, S., 2001: Feedforward Neural Networks: An Introduction. *Nonlinear Dynamical Systems: Feedforward Neural Network Perspectives*, I. Sandberg, Ed., John Wiley & Sons.

Hsu, W., and A. Murphy, 1986: The attributes diagram: A geometrical framework for assessing the quality of probability forecasts. *International Journal of Forecasting*, **2 (3)**, 285–293.

Hwang, Y., A. Clark, V. Lakshmanan, and S. Koch, 2015: Improved Nowcasts by Blending Extrapolation and Model Forecasts. *Weather and Forecasting*, **30 (5)**, 1201–1217.

Jeffreys, H., 1946: An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London A: Mathematical, Physical, and Engineering Sciences*, **186 (1007)**, 451–461.

Jesson, M., M. Haines, N. Singh, M. Sterling, and I. Taylor, 2013: Numerical and physical simulation of a thunderstorm downburst. *Asia-Pacific Conference on Wind Engineering*, Chennai, India, International Association for Wind Engineering.

Jewell, R., and J. Brimelow, 2009: Evaluation of Alberta hail growth model using severe hail proximity soundings from the United States. *Weather and Forecasting*, **24 (6)**, 1592–1609.

Johns, R., 1993: Meteorological conditions associated with bow echo development in convective storms. *Weather and Forecasting*, **8 (2)**, 294–299.

Karstens, C., T. Smith, K. Kuhlman, A. Clark, C. Ling, G. Stumpf, and L. Rothfusz, 2014: Prototype tool development for creating probabilistic hazard information for severe convective phenomena. *Symposium on Building a Weather-ready Nation*, Atlanta, Georgia, American Meteorological Society.

Key, J., J. Maslanik, and A. Schweiger, 1989: Classification of merged AVHRR and SMMR Arctic data with neural networks. *Photogrammetric Engineering and Remote Sensing*, **55 (9)**, 1331–1338.

Kitzmiller, D., W. McGovern, and R. Saffle, 1995: The WSR-88D severe weather potential algorithm. *Weather and Forecasting*, **10 (1)**, 141–159.

Kohavi, R., and G. John, 1997: Wrappers for feature subset selection. *Artificial Intelligence*, **97 (1)**, 273–324.

Lakshmanan, V., 2000: Using a genetic algorithm to tune a bounded weak echo region detection algorithm. *Journal of Applied Meteorology*, **39 (2)**, 222–230.

Lakshmanan, V., B. Herzog, and D. Kingfield, 2015: A Method for Extracting Postevent Storm Tracks. *Journal of Applied Meteorology and Climatology*, **54 (2)**, 451–462.

Lakshmanan, V., K. Hondl, and R. Rabin, 2009: An efficient, general-purpose technique for identifying storm cells in geospatial images. *Journal of Atmospheric and Oceanic Technology*, **26 (3)**, 523–537.

Lakshmanan, V., C. Karstens, J. Krause, and L. Tang, 2014: Quality control of weather radar data using polarimetric variables. *Journal of Atmospheric and Oceanic Technology*, **31 (6)**, 1234–1249.

Lakshmanan, V., and T. Smith, 2010: Evaluating a Storm Tracking Algorithm. *26th Conference on Interactive Information Processing Systems*, Atlanta, GA, American Meteorological Society.

Lakshmanan, V., T. Smith, K. Hondl, G. Stumpf, and A. Witt, 2006: A real-time, three-dimensional, rapidly updating, heterogeneous radar merger technique for reflectivity, velocity, and derived products. *Weather and Forecasting*, **21 (5)**, 802–823.

Lakshmanan, V., T. Smith, G. Stumpf, and K. Hondl, 2007: The Warning Decision Support System – Integrated Information. *Weather and Forecasting*, **22 (3)**, 596–612.

Lakshmanan, V., and A. Witt, 1997: A fuzzy logic approach to detecting severe updrafts. *AI Applications*, **11 (1)**, 1–12.

Lakshmanan, V., J. Zhang, and K. Howard, 2010: A technique to censor biological echoes in radar reflectivity data. *Journal of Applied Meteorology and Climatology*, **49 (3)**, 453–462.

Lin, J., 1991: Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, **37 (1)**, 145–151.

Luna-Herrera, J., G. Martinez-Cabrera, R. Parra-Maldonado, J. Enciso-Moreno, J. Torres-Lopez, F. Quesada-Pascual, R. Delgadillo-Polanco, and S. Franzblau, 2003: Use of receiver operating characteristic curves to assess the performance of a microdilution assay for determination of drug susceptibility of clinical isolates of Mycobacterium tuberculosis. *European Journal of Clinical Microbiology and Infectious Diseases*, **22 (1)**, 21–27.

Malone, T., 1955: Application of statistical methods in weather prediction. *Proceedings of the National Academy of Sciences*, **41 (11)**, 806–815.

Mansouryar, M., and A. Hedayati, 2012: Smoothing Via Iterative Averaging (SIA): A Basic Technique for Line Smoothing. *International Journal of Computer and Electrical Engineering*, **4 (3)**, 307–311.

Manzato, A., 2013: Hail in northeast italy: A neural network ensemble forecast using sounding-derived indices. *Weather and Forecasting*, **28 (1)**, 3–28.

Marzban, C., 2000: A neural network for tornado diagnosis: Managing local minima. *Neural Computing and Applications*, **9 (2)**, 133–141.

Marzban, C., and G. Stumpf, 1996: A neural network for tornado prediction based on Doppler radar-derived attributes. *Journal of Applied Meteorology*, **35 (5)**, 617–626.

Marzban, C., and G. Stumpf, 1998: A neural network for damaging wind prediction. *Weather and Forecasting*, **13 (1)**, 151–163.

Marzban, C., and A. Witt, 2001: A Bayesian neural network for severe-hail size prediction. *Weather and Forecasting*, **16 (5)**, 600–610.

McGovern, A., D. Gagne, J. Basara, T. Hamill, and D. Margolin, 2015: Solar energy prediction: an international contest to initiate interdisciplinary research on compelling meteorological problems. *Bulletin of the American Meteorological Society*, **96 (8)**, 1388–1395.

McGovern, A., D. Gagne, J. Williams, R. Brown, and J. Basara, 2014a: Tornadic supercell environments analyzed using surface and reanalysis data: a spatiotemporal relational data-mining approach. *Machine Learning*, **95 (1)**, 27–50.

McGovern, A., N. Hiers, M. Collier, D. Gagne, and R. Brown, 2008: Spatiotemporal Relational Probability Trees: An Introduction. *International Conference on Data Mining*, Pisa, Italy, Institute of Electrical and Electronics Engineers.

McGovern, A., D. Rosendahl, and R. Brown, 2014b: Toward Understanding Tornado Formation Through Spatiotemporal Data Mining. *Data Mining for Geoinformatics*, G. Cervone, J. Lin, and N. Waters, Eds., Springer.

McGovern, A., N. Troutman, R. Brown, J. Williams, and J. Abernethy, 2013: Enhanced spatiotemporal relational probability trees and forests. *Data Mining and Knowledge Discovery*, **26 (2)**, 398–433.

McNitt, J., J. Facundo, and J. O'Sullivan, 2008: Meteorological Assimilation Data Ingest System Transition Project Risk Reduction Activity. *24th Conference on Interactive Information Processing Systems*, New Orleans, LA, American Meteorological Society.

McPherson, R., and Coauthors, 2007: Statewide monitoring of the mesoscale environment: A technical update on the Oklahoma Mesonet. *Journal of Atmospheric and Oceanic Technology*, **24 (3)**, 301–321.

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. *Journal of Applied Meteorology and Climatology*, **54 (5)**, 1039–1059.

Mehdi, T., N. Bashardoost, and M. Ahmadi, 2011: Kernel smoothing for ROC curve and estimation for thyroid stimulating hormone. *International Journal of Public Health Research*, **Special Issue**, 239–242.

Mesinger, F., and Coauthors, 2006: North American Regional Reanalysis. *Bulletin of the American Meteorological Society*, **87 (3)**, 343–360.

Mitchell, T., 1997: *Machine Learning*. McGraw Hill.

Møller, M., 1993: A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, **6 (4)**, 525–533.

Muller, M., G. Tomlinson, T. Marrie, P. Tang, A. McGeer, D. Low, A. Detsky, and W. Gold, 2005: Can routine laboratory tests discriminate between severe acute respiratory syndrome and other causes of community-acquired pneumonia? *Clinical Infectious Diseases*, **40 (8)**, 1079–1086.

Murphy, A., 1993: What is a good forecast? An essay on the nature of goodness in weather forecasting. *Weather and Forecasting*, **8 (2)**, 281–293.

Niculescu-Mizil, A., and R. Caruana, 2005: Predicting good probabilities with supervised learning. *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, International Machine Learning Society.

Ortega, K., T. Smith, J. Zhang, C. Langston, Y. Qi, S. Stevens, and J. Tate, 2012: The multi-year reanalysis of remotely sensed storms (MYRORSS) project. *26th Conference on Severe Local Storms*, Nashville, TN, American Meteorological Society.

Pal, N., A. Mandal, S. Pal, J. Das, and V. Lakshmanan, 2006: Fuzzy rule-based approach for detection of bounded weak-echo regions in radar images. *Journal of Applied Meteorology and Climatology*, **45 (9)**, 1304–1312.

Park, H., A. Ryzhkov, D. Zrnić, and K.-E. Kim, 2009: The hydrometeor classification algorithm for the polarimetric WSR-88D: Description and application to an MCS. *Weather and Forecasting*, **24 (3)**, 730–748.

Platt, J., 1999: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, **10 (3)**, 61–74.

Quinlan, R., 1986: Induction of decision trees. *Machine Learning*, **1 (1)**, 81–106.

Reiter, E., 1991: Hybrid modeling in meteorological applications Part I: Concepts and approaches. *Meteorology and Atmospheric Physics*, **46 (1)**, 77–90.

Rico-Ramirez, M., and I. Cluckie, 2008: Classification of ground clutter and anomalous propagation using dual-polarization weather radar. *IEEE Transactions on Geoscience and Remote Sensing*, **46 (7)**, 1892–1904.

Riedmiller, M., and H. Braun, 1993: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *International Conference on Neural Networks*, San Francisco, CA, IEEE.

Rose, M., 1996: Downbursts. *National Weather Digest*, **21 (1)**, 11–17.

Saxen, T., 2002: Forecasting C-G lightning potential at WSMR. *Conference on Aviation, Range, and Aerospace Meteorology*, Portland, OR, American Meteorological Society.

Skamarock, W., and J. Klemp, 2008: A time-split nonhydrostatic atmospheric model for weather research and forecasting applications. *Journal of Computational Physics*, **227 (7)**, 3465–3485.

Smith, T., and Coauthors, 2016: Multi-radar Multi-sensor (MRMS) severe weather and aviation products: Initial operating capabilities. *Bulletin of the American Meteorological Society*, **Early online release**, 1–43.

Stumpf, G., A. Witt, E. Mitchell, P. Spencer, J. Johnson, M. Eilts, K. Thomas, and D. Burgess, 1998: The National Severe Storms Laboratory Mesocyclone Detection Algorithm for the WSR-88D. *Weather and Forecasting*, **13 (2)**, 304–326.

Supinie, T., A. McGovern, J. Williams, and J. Abernethy, 2009: Spatiotemporal relational random forests. *International Conference on Data Mining*, Miami, FL, Institute of Electrical and Electronics Engineers.

Tang, L., J. Zhang, Y. Wang, and K. Howard, 2011: Identification of Biological and Anomalous Propagation Echoes in Weather Radar Observations – An Imaging Processing Approach. *Proceedings of the 35th Conference on Radar Meteorology*, Pittsburgh, PA, American Meteorological Society.

Walker, S., and D. Duncan, 1967: Estimation of the probability of an event as a function of several independent variables. *Biometrika*, **54 (1-2)**, 167–179.

Wieler, J., 1986: Real-time automated detection of mesocyclones and tornadic vortex signatures. *Journal of Atmospheric and Oceanic Technology*, **3 (1)**, 98–113.

Williams, J., 2014: Using random forests to diagnose aviation turbulence. *Machine Learning*, **95 (1)**, 51–70.

Williams, J., D. Ahijevych, S. Dettling, and M. Steiner, 2008a: Combining observations and model data for short-term storm forecasting. *Remote Sensing Applications for Aviation Weather Hazard Detection and Decision Support*, San Diego, CA, International Society for Optics and Photonics.

Williams, J., R. Sharman, J. Craig, and G. Blackburn, 2008b: Remote detection and diagnosis of thunderstorm turbulence. *Remote Sensing Applications for Aviation Weather Hazard Detection and Decision Support*, San Diego, CA, International Society for Optics and Photonics.

Zipser, E., 1982: Use of a conceptual model of the life-cycle of mesoscale convective systems to improve very-short-range forecasts. *Nowcasting*, K. Browning, Ed., Academic Press.

Zou, H., and T. Hastie, 2005: Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67 (2)**, 301–320.

Zrnić, D., D. Burgess, and L. Hennington, 1985: Automatic detection of mesocyclonic shear with Doppler radar. *Journal of Atmospheric and Oceanic Technology*, **2 (4)**, 425–438.

Appendices

Appendix A

Model Setup for Experiment 1

A.1 Logistic Regression

Logistic regression is run with the parameters in Table A.1, using MATLAB's `glmfit`. These parameters force `glmfit` to run logistic regression (`link` = `logit`) for binary classification (`distr` = `binomial`), with the bias term (β_0 in Equation 4.1) included (`constant` = `on`). For each buffer distance and lead-time window, only one logistic model is trained, since there is only one choice for each parameter.

Parameter	**Values**
`distr`	`binomial`
`link`	`logit`
`constant`	`on`

Table A.1: Parameters for logistic regression, using MATLAB's `glmfit` (`http://www.mathworks.com/help/stats/glmfit.html`).

A.2 Logistic Regression with an Elastic Net

LREN is run with the parameters in Table A.2, using MATLAB's `lassoglm`. The parameters `distr` and `link` force `lassoglm` to run logistic regression for binary classification. Unlike `glmfit`, `lassoglm` does not need the parameter `constant`; it automatically includes β_0 in the model. `alpha` is α in Equation 4.3, which determines the balance between ridge ($\alpha = 0$) and lasso ($\alpha = 1$) regression. Our minimum value is 10^{-6} rather than 0, because `lassoglm` throws an error when `alpha` = 0. `numLambda` is the number of λ-values attempted (used in Equation 4.3 to determine the weight of the coefficient penalty). `lambdaRatio` is the ratio of the smallest to largest λ-values attempted (exact λ-values are chosen by `lassoglm`). The default in `lassoglm` is `lambdaRatio` = 10^{-4}, so we try five orders of magnitude, spaced evenly around the default. Finally, `relTol` is the stopping criterion. When the magnitudes of successive coefficient vectors ($\vec{\beta}$ in Equation 4.1) differ by $<$ `relTol`, the algorithm stops. Although the default in `lassoglm` is 10^{-4}, in our experience the algorithm does not converge when `relTol` is much less than 0.01.

For each set of parameters in Table A.2, `lassoglm` trains 50 models (one for each λ-value). Ideally we would keep the model with the highest AUC, but AUC calculations are very computationally expensive (see end of Section 6.1.1). Thus, we keep the model with the lowest deviance (Equation 4.2), which is the objective function for logistic regression.

For each buffer distance and lead-time window, 25 LREN models are trained (one for each of five `alpha` values and five `lambdaRatio` values).

Parameter	Values
`distr`	`binomial`
`link`	`logit`
`alpha`	10^{-6}, 0.25, 0.5, 0.75, 1
`numLambda`	50
`lambdaRatio`	10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}
`relTol`	0.01

Table A.2: Parameters for LREN, using MATLAB's `lassoglm` (`http://www.mathworks.com/help/stats/lassoglm.html`).

A.3 Feed-forward Neural Nets

FFNN is run with the parameters in Table A.3, using MATLAB's `patternnet`. Parameters are set in two steps. First, the neural-net object is created by calling `patternnet` with `hiddenSizes`, `trainFcn`, and `performFcn` as input arguments. Then, letting the object be `ffnn`, remaining parameters are set by calling `ffnn.trainParam.epochs = 1000`, `ffnn.trainParam.delta0 = 0.001`, etc.

`hiddenSizes` is a two-element vector, with the number of neurons in the first and second hidden layers respectively. Running neural nets with more than two hidden layers, or more than ~20 neurons for each of the two layers, is too computationally expensive. `trainFcn` is the backpropagation method; the two options stand for resilient backpropagation (Riedmiller and Braun, 1993) and scaled conjugate gradient (SCG) (Møller, 1993), respectively. `performFcn` is `crossentropy`, which is another word for deviance (Equation 4.2). `epochs` is the number of epochs, or number of times that each training example is presented to the neural net. The stopping criterion (that deviance on validation

examples increases between successive epochs) is almost always reached before 1000 epochs. `delta0` and `deltamax` are the initial and maximum update weights for backpropagation, respectively. `delt_dec` and `delt_inc` are values by which the update weight may be multiplied between successive epochs. These four parameters are valid only for resilient backpropagation, while `sigma` and `lambda` are valid only for SCG backpropagation. `sigma` and `lambda` (σ and λ respectively in Equation 20 of Riedmiller and Braun, 1993) work together to determine the update weight.

For each buffer distance and lead-time window, 800 FFNN models are trained (one for each of two backpropagation methods, 16 combinations of hidden-layer sizes, five values of `delt_dec` or `sigma`, and five values of `delt_inc` or `lambda`).

A.4 Random Forests

Random forests are run with the parameters in Table A.4, using MATLAB's `fitensemble`. Again, parameters are set in two steps. First, a decision-tree template is created by calling `templateTree` with the input arguments `type` = `classification`, `nVarToSample`, `minLeaf`, `splitCriterion`, and `mergeLeaves`. Let the result be `dtTemplate`. Then `fitensemble` is called with `method` = `bag` (to create a random forest rather than some other kind of ensemble), `learners` = `dtTemplate`, `type` = `classification`, and all parameters listed in Table A.4 but not used for `templateTree`.

`nVarToSample` is the number of predictors to sample randomly at each split (this sampling is called "feature-bagging" and described in Section 4.1.1.5). The default value for random forests is $\sqrt{N}$, where N is the number of predictors (~21 for $N = 431$). Thus, we try values on the same order of magnitude, one below, and one above. `minLeaf` is the minimum number of training examples at a leaf node (one of three stopping criteria discussed in Section 4.1.1.5).

Parameter	Values
`hiddenSizes`	$\{5,5\}$; $\{5,10\}$; $\{5,15\}$; $\{5,20\}$; $\{10,5\}$; $\{10,10\}$; $\{10,15\}$; $\{10,20\}$; $\{15,5\}$; $\{15,10\}$; $\{15,15\}$; $\{15,20\}$; $\{20,5\}$; $\{20,10\}$; $\{20,15\}$; $\{20,20\}$
`trainFcn`	`trainrp`, `trainscg`
`performFcn`	`crossentropy`
`epochs`	0.001
`delta0`	10^{-3}
`deltamax`	1000
`delt_dec`	0.001, 0.01, 0.1, 0.5, 0.9
`delt_inc`	1.1, 1.5, 10, 100, 1000
`sigma`	5×10^{-7}, 5×10^{-6}, 5×10^{-5}, 5×10^{-4}, 5×10^{-3}
`lambda`	5×10^{-9}, 5×10^{-8}, 5×10^{-7}, 5×10^{-6}, 5×10^{-5}

Table A.3: Parameters for FFNN, using MATLAB's `patternnet` (`http://www.mathworks.com/help/nnet/ref/patternnet.html`).

Parameter	Values
nVarToSample	1, 5, 10, 25, 50, 75, 100
minLeaf	5, 10, 25, 50, 75
splitCriterion	deviance
mergeLeaves	on
nLearn	500
resample	on
fResample	0.05, 0.20, 0.40, 0.60, 0.80, 1.00
replace	on

Table A.4: Parameters for random forests, using MATLAB's `fitensemble` (`http://www.mathworks.com/help/stats/fitensemble.html`).

`splitCriterion` is the objective function (`deviance` is interpreted as information gain [Equation 4.7] by `fitensemble`). `mergeLeaves` causes leaf nodes to be merged if they originate from the same parent node and, when applied to validation data, lead to negative information gain. `nLearn` is the number of decision trees in the ensemble. The ensemble is regularized after training (the number of trees leading to the lowest deviance[1] on validation data is chosen), and this number of trees is almost always < 500. `resample` = `on` means that training examples are resampled for each tree (this is called "tree-bagging" and described in Section 4.1.1.5). `fResample` is the fraction of training examples resampled for each tree, and `replace` = `on` ensures that sampling is done with replacement.

For each buffer distance and lead-time window, 210 random forests are trained (one for each of seven `nVarToSample` values, five `minLeaf` values, and six `fResample` values).

[1]Ideally we would keep the model with the highest AUC, but AUC calculations are very computationally expensive (see end of Section 6.1.1).

A.5 Ensembles of Gradient-boosted Trees

GBT ensembles are run with the parameters in Table A.5, using MATLAB's `fitensemble`. As for random forests, the parameters `nVarToSample`, `minLeaf`, `splitCriterion`, and `mergeLeaves` are sent to `templateTree`; the others are sent to `fitensemble`.

`minLeaf`, `splitCriterion`, `mergeLeaves`, `nLearn`, `resample`, and `replace` all have the same meaning and values as for random forests. Larger values of `nVarToSample` are tried, because in our experience GBT ensembles perform badly with small values. Similarly, smaller values of `fResample` are tried, because we find that GBT ensembles perform badly with large values. Ensembles are regularized in the same way as random forests. `method` is the boosting method (`AdaBoostM1` is AdaBoost [Section 4.1.1.6] for binary classification). For a `learnRate` of α, `fitensemble` fits every new tree to αr_{iq} (α times the pseudo-residual) rather than just r_{iq} (see Equation 4.10). `learnRate` varies from $(0, 1]$; the algorithm learns more quickly for larger values. In our experience (consistent with Friedman, 2001), values $\gtrsim 0.2$ lead to overfitting.

For each buffer distance and lead-time window, 750 GBT ensembles are trained (one for each of five `nVarsToSample` values, five `minLeaf` values, six `fResample` values, and five `learnRate` values).

A.6 Isotonic Regression

Isotonic regression is run with MATLAB's `lsqisotonic`[2], which requires no input parameters beyond the training data.

[2] `http://www.mathworks.com/matlabcentral/fileexchange/47196-graph-based-clustering-and-data-visualization-algorithms/content/improve_JP/toolbox_imp_JP/lsqisotonic.m`

Parameter	Values
`nVarToSample`	10, 25, 50, 100, 431
`minLeaf`	5, 10, 25, 50, 75
`splitCriterion`	`deviance`
`mergeLeaves`	`on`
`method`	`AdaBoostM1`
`nLearn`	`500`
`resample`	`on`
`fResample`	0.05, 0.10, 0.15, 0.20, 0.25, 0.30
`replace`	`on`
`learnRate`	0.01, 0.05, 0.10, 0.15, 0.20

Table A.5: Parameters for GBT ensembles, using MATLAB's `fitensemble`.

Appendix B

Selected Models for Experiment 1

Selected models for Experiment 1 (via the procedure in Section 6.1.1) are listed in Tables B.1-B.15. For a given buffer distance and lead-time window, if more than 20 models have been selected, only the top 20 are shown.

Base Model	AUC	Parameters
Random forest	0.9962	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 10
GBT ensemble	0.9934	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.2
GBT ensemble	0.9932	`fResample` = 0.05; `nVarToSample` = 431; `minLeaf` = 5; `learnRate` = 0.2
GBT ensemble	0.9929	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 75; `learnRate` = 0.15
Random forest	0.9927	`fResample` = 0.60; `nVarToSample` = 25; `minLeaf` = 25
GBT ensemble	0.9926	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 25; `learnRate` = 0.15

Continued on next page

Table B.1 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9921	`fResample` = 0.05; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.2
GBT ensemble	0.9920	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 5; `learnRate` = 0.05
GBT ensemble	0.9919	`fResample` = 0.30; `nVarToSample` = 50; `minLeaf` = 25; `learnRate` = 0.05
FFNN	0.9917	$\{5, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-6}
GBT ensemble	0.9916	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 25; `learnRate` = 0.01
GBT ensemble	0.9914	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.01
FFNN	0.9909	$\{5, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-6}; `lambda` = 5×10^{-5}
GBT ensemble	0.9907	`fResample` = 0.25; `nVarToSample` = 50; `minLeaf` = 25; `learnRate` = 0.01
FFNN	0.9907	$\{5, 5\}$ neurons; `trainrp`; `delt_dec` = 0.9; `delt_inc` = 1.5
Random forest	0.9906	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 25
FFNN	0.9904	$\{5, 15\}$ neurons; `trainscg`; `sigma` = 5×10^{-6}; `lambda` = 5×10^{-5}
Random forest	0.9904	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 25

Continued on next page

Table B.1 – continued from previous page

Base Model	AUC	Parameters
FFNN	0.9903	$\{20, 20\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-6}$; `lambda` $= 5 \times 10^{-6}$
FFNN	0.9902	$\{10, 20\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-5}$; `lambda` $= 5 \times 10^{-8}$

Table B.1: Selected models for buffer distance of 0 km and lead time of $[0, 15]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9687	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.1
GBT ensemble	0.9674	`fResample` = 0.25; `nVarToSample` = 431; `minLeaf` = 10; `learnRate` = 0.01
Random forest	0.9652	`fResample` = 0.40; `nVarToSample` = 25; `minLeaf` = 10
GBT ensemble	0.9642	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.01
Random forest	0.9639	`fResample` = 0.40; `nVarToSample` = 75; `minLeaf` = 50
GBT ensemble	0.9636	`fResample` = 0.30; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.1
Random forest	0.9631	`fResample` = 0.60; `nVarToSample` = 100; `minLeaf` = 75
Random forest	0.9631	`fResample` = 0.60; `nVarToSample` = 50; `minLeaf` = 25
GBT ensemble	0.9631	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 5; `learnRate` = 0.1
GBT ensemble	0.9629	`fResample` = 0.15; `nVarToSample` = 50; `minLeaf` = 25; `learnRate` = 0.15
GBT ensemble	0.9629	`fResample` = 0.30; `nVarToSample` = 100; `minLeaf` = 5; `learnRate` = 0.05
GBT ensemble	0.9626	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.1

Continued on next page

Table B.2 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9624	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 25; `learnRate` = 0.15
Random forest	0.9622	`fResample` = 0.80; `nVarToSample` = 100; `minLeaf` = 50
Random forest	0.9620	`fResample` = 0.60; `nVarToSample` = 5; `minLeaf` = 5
Random forest	0.9619	`fResample` = 1.00; `nVarToSample` = 50; `minLeaf` = 25
GBT ensemble	0.9613	`fResample` = 0.25; `nVarToSample` = 10; `minLeaf` = 5; `learnRate` = 0.1
Random forest	0.9608	`fResample` = 1.00; `nVarToSample` = 75; `minLeaf` = 50
Random forest	0.9606	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 10
FFNN	0.9606	$\{20, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-7}; `lambda` = 5×10^{-6}

Table B.2: Selected models for buffer distance of 0 km and lead time of $[15, 30]$ minutes.

Base Model	AUC	Parameters
Random forest	0.9470	`fResample` = 0.60; `nVarToSample` = 10; `minLeaf` = 10
GBT ensemble	0.9450	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.05
Random forest	0.9449	`fResample` = 0.80; `nVarToSample` = 75; `minLeaf` = 75
GBT ensemble	0.9446	`fResample` = 0.15; `nVarToSample` = 50; `minLeaf` = 50; `learnRate` = 0.01
GBT ensemble	0.9443	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 25; `learnRate` = 0.05
GBT ensemble	0.9441	`fResample` = 0.20; `nVarToSample` = 10; `minLeaf` = 25; `learnRate` = 0.05
Random forest	0.9441	`fResample` = 0.40; `nVarToSample` = 100; `minLeaf` = 25
Random forest	0.9434	`fResample` = 0.40; `nVarToSample` = 25; `minLeaf` = 5
GBT ensemble	0.9431	`fResample` = 0.10; `nVarToSample` = 50; `minLeaf` = 10; `learnRate` = 0.1
Random forest	0.9427	`fResample` = 0.20; `nVarToSample` = 1; `minLeaf` = 5
GBT ensemble	0.9426	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9426	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.1

Continued on next page

Table B.3 – continued from previous page

Base Model	AUC	Parameters
FFNN	0.9424	$\{20, 5\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-4}$; `lambda` $= 5 \times 10^{-5}$
GBT ensemble	0.9422	`fResample` $= 0.30$; `nVarToSample` $= 10$; `minLeaf` $= 10$; `learnRate` $= 0.01$
GBT ensemble	0.9421	`fResample` $= 0.25$; `nVarToSample` $= 10$; `minLeaf` $= 75$; `learnRate` $= 0.01$
GBT ensemble	0.9420	`fResample` $= 0.20$; `nVarToSample` $= 50$; `minLeaf` $= 50$; `learnRate` $= 0.2$
GBT ensemble	0.9419	`fResample` $= 0.25$; `nVarToSample` $= 100$; `minLeaf` $= 75$; `learnRate` $= 0.1$
Random forest	0.9418	`fResample` $= 0.80$; `nVarToSample` $= 25$; `minLeaf` $= 5$
LREN	0.9418	`alpha` $= 0.50$, `lambdaRatio` $= 10^{-2}$
GBT ensemble	0.9418	`fResample` $= 0.10$; `nVarToSample` $= 10$; `minLeaf` $= 25$; `learnRate` $= 0.1$

Table B.3: Selected models for buffer distance of 0 km and lead time of $[30, 45]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9440	`fResample` = 0.30; `nVarToSample` = 50; `minLeaf` = 5; `learnRate` = 0.15
GBT ensemble	0.9414	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.05
GBT ensemble	0.9410	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 75; `learnRate` = 0.05
Random forest	0.9406	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 50
GBT ensemble	0.9393	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.1
GBT ensemble	0.9388	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 75; `learnRate` = 0.01
GBT ensemble	0.9377	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.1
Random forest	0.9375	`fResample` = 0.60; `nVarToSample` = 100; `minLeaf` = 5
GBT ensemble	0.9372	`fResample` = 0.15; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.1
GBT ensemble	0.9366	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 50; `learnRate` = 0.2
GBT ensemble	0.9361	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 75; `learnRate` = 0.15
GBT ensemble	0.9358	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 5; `learnRate` = 0.15

Continued on next page

Table B.4 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9354	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 25; `learnRate` = 0.01
Random forest	0.9350	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 10
GBT ensemble	0.9349	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.2
Random forest	0.9348	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 10
Random forest	0.9346	`fResample` = 0.60; `nVarToSample` = 100; `minLeaf` = 10

Table B.4: Selected models for buffer distance of 0 km and lead time of $[45, 60]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9371	`fResample` = 0.25; `nVarToSample` = 431; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9334	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 25; `learnRate` = 0.1
Random forest	0.9316	`fResample` = 0.80; `nVarToSample` = 100; `minLeaf` = 10
GBT ensemble	0.9309	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.1
GBT ensemble	0.9293	`fResample` = 0.05; `nVarToSample` = 10; `minLeaf` = 50; `learnRate` = 0.05
Random forest	0.9289	`fResample` = 0.60; `nVarToSample` = 25; `minLeaf` = 25
GBT ensemble	0.9276	`fResample` = 0.20; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.1

Table B.5: Selected models for buffer distance of 0 km and lead time of $[60, 90]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9662	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 25; `learnRate` = 0.1
GBT ensemble	0.9642	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 5; `learnRate` = 0.05
GBT ensemble	0.9624	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 50; `learnRate` = 0.1
GBT ensemble	0.9611	`fResample` = 0.25; `nVarToSample` = 10; `minLeaf` = 50; `learnRate` = 0.01
GBT ensemble	0.9611	`fResample` = 0.10; `nVarToSample` = 431; `minLeaf` = 50; `learnRate` = 0.05
GBT ensemble	0.9611	`fResample` = 0.10; `nVarToSample` = 10; `minLeaf` = 5; `learnRate` = 0.01
GBT ensemble	0.9607	`fResample` = 0.05; `nVarToSample` = 431; `minLeaf` = 50; `learnRate` = 0.05
GBT ensemble	0.9594	`fResample` = 0.30; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.1
FFNN	0.9582	$\{10, 15\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-7}
Random forest	0.9580	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 25
GBT ensemble	0.9579	`fResample` = 0.25; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.1
Random forest	0.9577	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 5

Continued on next page

Table B.6 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9576	`fResample` = 0.20; `nVarToSample` = 10; `minLeaf` = 25; `learnRate` = 0.05
FFNN	0.9571	$\{10, 20\}$ neurons; `trainrp`; `delt_dec` = 0.1; `delt_inc` = 1.1
LREN	0.9568	`alpha` = 0.75, `lambdaRatio` = 10^{-6}
GBT ensemble	0.9567	`fResample` = 0.30; `nVarToSample` = 25; `minLeaf` = 5; `learnRate` = 0.1
Random forest	0.9566	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 50
Random forest	0.9564	`fResample` = 0.80; `nVarToSample` = 5; `minLeaf` = 25
FFNN	0.9563	$\{20, 20\}$ neurons; `trainrp`; `delt_dec` = 0.1; `delt_inc` = 1.1

Table B.6: Selected models for buffer distance of 5 km and lead time of $[0, 15]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9482	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.01
GBT ensemble	0.9480	`fResample` = 0.30; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.01
Random forest	0.9462	`fResample` = 0.40; `nVarToSample` = 50; `minLeaf` = 5
GBT ensemble	0.9460	`fResample` = 0.30; `nVarToSample` = 50; `minLeaf` = 10; `learnRate` = 0.05
GBT ensemble	0.9449	`fResample` = 0.10; `nVarToSample` = 50; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9446	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9442	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9440	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 50; `learnRate` = 0.05
Random forest	0.9438	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 5
GBT ensemble	0.9433	`fResample` = 0.30; `nVarToSample` = 10; `minLeaf` = 50; `learnRate` = 0.05
Random forest	0.9426	`fResample` = 0.20; `nVarToSample` = 50; `minLeaf` = 25
FFNN	0.9424	$\{20, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-9}

Continued on next page

Table B.7 – continued from previous page

Base Model	AUC	Parameters
Random forest	0.9422	`fResample` = 0.20; `nVarToSample` = 5; `minLeaf` = 10
GBT ensemble	0.9418	`fResample` = 0.30; `nVarToSample` = 50; `minLeaf` = 75; `learnRate` = 0.01
FFNN	0.9418	$\{20, 20\}$ neurons; `trainrp`; `delt_dec` = 0.5; `delt_inc` = 1.5
GBT ensemble	0.9416	`fResample` = 0.15; `nVarToSample` = 431; `minLeaf` = 25; `learnRate` = 0.01
FFNN	0.9414	$\{15, 10\}$ neurons; `trainrp`; `delt_dec` = 0.1; `delt_inc` = 1.1
Random forest	0.9407	`fResample` = 0.80; `nVarToSample` = 100; `minLeaf` = 50
Random forest	0.9403	`fResample` = 1.00; `nVarToSample` = 100; `minLeaf` = 25
GBT ensemble	0.9403	`fResample` = 0.15; `nVarToSample` = 431; `minLeaf` = 10; `learnRate` = 0.05

Table B.7: Selected models for buffer distance of 5 km and lead time of $[15, 30]$ minutes.

Base Model	AUC	Parameters
FFNN	0.9265	$\{5, 5\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-5}$; `lambda` $= 5 \times 10^{-8}$
GBT ensemble	0.9259	`fResample` $= 0.25$; `nVarToSample` $= 50$; `minLeaf` $= 50$; `learnRate` $= 0.01$
FFNN	0.9255	$\{15, 5\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-5}$; `lambda` $= 5 \times 10^{-5}$
Random forest	0.9252	`fResample` $= 0.40$; `nVarToSample` $= 25$; `minLeaf` $= 10$
GBT ensemble	0.9251	`fResample` $= 0.30$; `nVarToSample` $= 25$; `minLeaf` $= 10$; `learnRate` $= 0.05$
GBT ensemble	0.9249	`fResample` $= 0.20$; `nVarToSample` $= 25$; `minLeaf` $= 25$; `learnRate` $= 0.1$
Random forest	0.9245	`fResample` $= 1.00$; `nVarToSample` $= 10$; `minLeaf` $= 50$
GBT ensemble	0.9244	`fResample` $= 0.10$; `nVarToSample` $= 25$; `minLeaf` $= 75$; `learnRate` $= 0.1$
GBT ensemble	0.9240	`fResample` $= 0.30$; `nVarToSample` $= 25$; `minLeaf` $= 50$; `learnRate` $= 0.1$
Random forest	0.9239	`fResample` $= 1.00$; `nVarToSample` $= 25$; `minLeaf` $= 50$
GBT ensemble	0.9239	`fResample` $= 0.30$; `nVarToSample` $= 10$; `minLeaf` $= 75$; `learnRate` $= 0.15$
FFNN	0.9238	$\{20, 15\}$ neurons; `trainrp`; `delt_dec` $= 0.01$; `delt_inc` $= 1.1$

Continued on next page

Table B.8 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9237	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.1
GBT ensemble	0.9236	`fResample` = 0.15; `nVarToSample` = 50; `minLeaf` = 75; `learnRate` = 0.1
Random forest	0.9235	`fResample` = 1.00; `nVarToSample` = 75; `minLeaf` = 25
Random forest	0.9233	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 50
GBT ensemble	0.9232	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 5; `learnRate` = 0.1
Random forest	0.9229	`fResample` = 0.80; `nVarToSample` = 5; `minLeaf` = 10
Random forest	0.9229	`fResample` = 0.40; `nVarToSample` = 10; `minLeaf` = 5
Random forest	0.9229	`fResample` = 0.40; `nVarToSample` = 50; `minLeaf` = 10

Table B.8: Selected models for buffer distance of 5 km and lead time of $[30, 45]$ minutes.

Base Model	AUC	Parameters
Random forest	0.9227	`fResample` = 0.60; `nVarToSample` = 25; `minLeaf` = 5
Random forest	0.9219	`fResample` = 0.40; `nVarToSample` = 5; `minLeaf` = 5
GBT ensemble	0.9210	`fResample` = 0.15; `nVarToSample` = 431; `minLeaf` = 50; `learnRate` = 0.05
Random forest	0.9201	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 25
Random forest	0.9194	`fResample` = 0.05; `nVarToSample` = 100; `minLeaf` = 5
Random forest	0.9191	`fResample` = 0.60; `nVarToSample` = 100; `minLeaf` = 25
GBT ensemble	0.9186	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.15
LREN	0.9185	`alpha` = 0.75, `lambdaRatio` = 10^{-4}
FFNN	0.9184	$\{20, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-7}
GBT ensemble	0.9178	`fResample` = 0.05; `nVarToSample` = 25; `minLeaf` = 75; `learnRate` = 0.15
GBT ensemble	0.9177	`fResample` = 0.25; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.01
LREN	0.9177	`alpha` = 0.75, `lambdaRatio` = 10^{-6}
Random forest	0.9172	`fResample` = 0.40; `nVarToSample` = 75; `minLeaf` = 25

Continued on next page

Table B.9 – continued from previous page

Base Model	AUC	Parameters
Random forest	0.9172	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 25
Random forest	0.9165	`fResample` = 1.00; `nVarToSample` = 50; `minLeaf` = 10
GBT ensemble	0.9164	`fResample` = 0.25; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.1
FFNN	0.9163	$\{10, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-6}
Random forest	0.9162	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 5
Random forest	0.9161	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 50
GBT ensemble	0.9161	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 5; `learnRate` = 0.05

Table B.9: Selected models for buffer distance of 5 km and lead time of $[45, 60]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9111	`fResample` = 0.30; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.01
GBT ensemble	0.9095	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 25; `learnRate` = 0.05
GBT ensemble	0.9065	`fResample` = 0.15; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.05
FFNN	0.9065	$\{20, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-6}
GBT ensemble	0.9063	`fResample` = 0.25; `nVarToSample` = 25; `minLeaf` = 75; `learnRate` = 0.05
Random forest	0.9059	`fResample` = 0.05; `nVarToSample` = 100; `minLeaf` = 10
GBT ensemble	0.9059	`fResample` = 0.30; `nVarToSample` = 10; `minLeaf` = 5; `learnRate` = 0.1
FFNN	0.9047	$\{15, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-7}; `lambda` = 5×10^{-8}
FFNN	0.9046	$\{10, 15\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-7}
GBT ensemble	0.9042	`fResample` = 0.15; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.05
GBT ensemble	0.9040	`fResample` = 0.20; `nVarToSample` = 10; `minLeaf` = 50; `learnRate` = 0.05
GBT ensemble	0.9027	`fResample` = 0.25; `nVarToSample` = 100; `minLeaf` = 25; `learnRate` = 0.05

Continued on next page

Table B.10 – continued from previous page

Base Model	AUC	Parameters
Random forest	0.9025	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 5
FFNN	0.9024	$\{15, 20\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-5}
GBT ensemble	0.9023	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.05
FFNN	0.9021	$\{15, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-7}
GBT ensemble	0.9021	`fResample` = 0.25; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.05
GBT ensemble	0.9016	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 75; `learnRate` = 0.1

Table B.10: Selected models for buffer distance of 5 km and lead time of $[60, 90]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9449	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 5; `learnRate` = 0.15
LREN	0.9428	`alpha` = 0.50, `lambdaRatio` = 10^{-5}
GBT ensemble	0.9421	`fResample` = 0.15; `nVarToSample` = 100; `minLeaf` = 5; `learnRate` = 0.05
Random forest	0.9410	`fResample` = 1.00; `nVarToSample` = 100; `minLeaf` = 50
GBT ensemble	0.9401	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 50; `learnRate` = 0.05
FFNN	0.9394	$\{15, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-5}
Random forest	0.9392	`fResample` = 0.80; `nVarToSample` = 100; `minLeaf` = 50
GBT ensemble	0.9391	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.1
FFNN	0.9387	$\{10, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-7}; `lambda` = 5×10^{-7}
Random forest	0.9384	`fResample` = 0.60; `nVarToSample` = 50; `minLeaf` = 25
GBT ensemble	0.9383	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 25; `learnRate` = 0.05
GBT ensemble	0.9382	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 5; `learnRate` = 0.2
FFNN	0.9380	$\{15, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-7}; `lambda` = 5×10^{-8}

Continued on next page

Table B.11 – continued from previous page

Base Model	AUC	Parameters
Random forest	0.9379	`fResample` = 1.00; `nVarToSample` = 50; `minLeaf` = 25
GBT ensemble	0.9378	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.05
LREN	0.9378	`alpha` = 0.25, `lambdaRatio` = 10^{-6}
Random forest	0.9378	`fResample` = 0.80; `nVarToSample` = 5; `minLeaf` = 10
Random forest	0.9376	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 5
GBT ensemble	0.9372	`fResample` = 0.20; `nVarToSample` = 431; `minLeaf` = 75; `learnRate` = 0.1
GBT ensemble	0.9371	`fResample` = 0.10; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.05

Table B.11: Selected models for buffer distance of 10 km and lead time of $[0, 15]$ minutes.

Base Model	AUC	Parameters
LREN	0.9317	`alpha` = 1.00, `lambdaRatio` = 10^{-6}
Random forest	0.9302	`fResample` = 1.00; `nVarToSample` = 25; `minLeaf` = 5
LREN	0.9300	`alpha` = 0.75, `lambdaRatio` = 10^{-2}
GBT ensemble	0.9283	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.01
FFNN	0.9283	$\{15, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-7}
GBT ensemble	0.9282	`fResample` = 0.25; `nVarToSample` = 25; `minLeaf` = 25; `learnRate` = 0.01
Random forest	0.9280	`fResample` = 0.40; `nVarToSample` = 25; `minLeaf` = 25
Random forest	0.9278	`fResample` = 0.40; `nVarToSample` = 10; `minLeaf` = 5
Random forest	0.9276	`fResample` = 0.60; `nVarToSample` = 10; `minLeaf` = 5
GBT ensemble	0.9276	`fResample` = 0.25; `nVarToSample` = 50; `minLeaf` = 50; `learnRate` = 0.05
GBT ensemble	0.9272	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.01
FFNN	0.9269	$\{15, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-7}; `lambda` = 5×10^{-9}
GBT ensemble	0.9265	`fResample` = 0.15; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.05

Continued on next page

Table B.12 – continued from previous page

Base Model	AUC	Parameters
Random forest	0.9265	`fResample` = 0.80; `nVarToSample` = 25; `minLeaf` = 10
GBT ensemble	0.9260	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 75; `learnRate` = 0.1
Random forest	0.9257	`fResample` = 0.40; `nVarToSample` = 10; `minLeaf` = 10
GBT ensemble	0.9255	`fResample` = 0.30; `nVarToSample` = 10; `minLeaf` = 5; `learnRate` = 0.2
GBT ensemble	0.9254	`fResample` = 0.25; `nVarToSample` = 431; `minLeaf` = 5; `learnRate` = 0.2
FFNN	0.9252	$\{5, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-9}
FFNN	0.9251	$\{20, 20\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-5}

Table B.12: Selected models for buffer distance of 10 km and lead time of $[15, 30]$ minutes.

Base Model	AUC	Parameters
FFNN	0.9202	$\{10, 10\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-5}$; `lambda` $= 5 \times 10^{-6}$
Random forest	0.9167	`fResample` $= 0.60$; `nVarToSample` $= 75$; `minLeaf` $= 10$
GBT ensemble	0.9161	`fResample` $= 0.25$; `nVarToSample` $= 25$; `minLeaf` $= 10$; `learnRate` $= 0.05$
Random forest	0.9158	`fResample` $= 0.80$; `nVarToSample` $= 5$; `minLeaf` $= 50$
FFNN	0.9154	$\{20, 5\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-4}$; `lambda` $= 5 \times 10^{-9}$
Random forest	0.9152	`fResample` $= 1.00$; `nVarToSample` $= 5$; `minLeaf` $= 5$
GBT ensemble	0.9138	`fResample` $= 0.25$; `nVarToSample` $= 100$; `minLeaf` $= 50$; `learnRate` $= 0.05$
GBT ensemble	0.9130	`fResample` $= 0.15$; `nVarToSample` $= 100$; `minLeaf` $= 5$; `learnRate` $= 0.1$
Random forest	0.9127	`fResample` $= 0.60$; `nVarToSample` $= 5$; `minLeaf` $= 25$
FFNN	0.9126	$\{10, 15\}$ neurons; `trainrp`; `delt_dec` $= 0.5$; `delt_inc` $= 1.1$
GBT ensemble	0.9126	`fResample` $= 0.10$; `nVarToSample` $= 100$; `minLeaf` $= 75$; `learnRate` $= 0.05$
Random forest	0.9123	`fResample` $= 0.80$; `nVarToSample` $= 100$; `minLeaf` $= 25$

Continued on next page

Table B.13 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9122	`fResample` = 0.30; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.05
GBT ensemble	0.9120	`fResample` = 0.30; `nVarToSample` = 100; `minLeaf` = 10; `learnRate` = 0.1
GBT ensemble	0.9117	`fResample` = 0.10; `nVarToSample` = 25; `minLeaf` = 50; `learnRate` = 0.01
GBT ensemble	0.9113	`fResample` = 0.30; `nVarToSample` = 100; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9112	`fResample` = 0.15; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.01
GBT ensemble	0.9108	`fResample` = 0.15; `nVarToSample` = 50; `minLeaf` = 25; `learnRate` = 0.05
GBT ensemble	0.9106	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 50; `learnRate` = 0.05
Random forest	0.9105	`fResample` = 0.80; `nVarToSample` = 50; `minLeaf` = 10

Table B.13: Selected models for buffer distance of 10 km and lead time of $[30, 45]$ minutes.

Base Model	AUC	Parameters
GBT ensemble	0.9155	`fResample` = 0.20; `nVarToSample` = 10; `minLeaf` = 75; `learnRate` = 0.05
GBT ensemble	0.9111	`fResample` = 0.30; `nVarToSample` = 50; `minLeaf` = 10; `learnRate` = 0.15
Random forest	0.9110	`fResample` = 0.60; `nVarToSample` = 10; `minLeaf` = 25
Random forest	0.9087	`fResample` = 0.80; `nVarToSample` = 75; `minLeaf` = 5
FFNN	0.9087	$\{20, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-5}
FFNN	0.9077	$\{5, 20\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-6}
FFNN	0.9073	$\{15, 10\}$ neurons; `trainrp`; `delt_dec` = 0.5; `delt_inc` = 1.1
Random forest	0.9071	`fResample` = 0.40; `nVarToSample` = 1; `minLeaf` = 10
Random forest	0.9068	`fResample` = 0.60; `nVarToSample` = 75; `minLeaf` = 25
LREN	0.9068	`alpha` = 0.75, `lambdaRatio` = 10^{-5}
GBT ensemble	0.9066	`fResample` = 0.20; `nVarToSample` = 100; `minLeaf` = 25; `learnRate` = 0.15
FFNN	0.9064	$\{20, 15\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-8}
FFNN	0.9063	$\{10, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-5}; `lambda` = 5×10^{-6}

Continued on next page

Table B.14 – continued from previous page

Base Model	AUC	Parameters
GBT ensemble	0.9060	`fResample` = 0.15; `nVarToSample` = 431; `minLeaf` = 5; `learnRate` = 0.1
GBT ensemble	0.9058	`fResample` = 0.15; `nVarToSample` = 50; `minLeaf` = 5; `learnRate` = 0.05
Random forest	0.9058	`fResample` = 0.80; `nVarToSample` = 1; `minLeaf` = 25
Random forest	0.9057	`fResample` = 0.80; `nVarToSample` = 100; `minLeaf` = 10

Table B.14: Selected models for buffer distance of 10 km and lead time of $[45, 60]$ minutes.

Base Model	**AUC**	**Parameters**
Random forest	0.8912	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 50
GBT ensemble	0.8883	`fResample` = 0.30; `nVarToSample` = 431; `minLeaf` = 50; `learnRate` = 0.01
Random forest	0.8867	`fResample` = 1.00; `nVarToSample` = 10; `minLeaf` = 10
GBT ensemble	0.8852	`fResample` = 0.15; `nVarToSample` = 25; `minLeaf` = 10; `learnRate` = 0.05
Random forest	0.8851	`fResample` = 0.05; `nVarToSample` = 10; `minLeaf` = 10
GBT ensemble	0.8825	`fResample` = 0.25; `nVarToSample` = 10; `minLeaf` = 10; `learnRate` = 0.01
Random forest	0.8823	`fResample` = 0.20; `nVarToSample` = 25; `minLeaf` = 10

Table B.15: Selected models for buffer distance of 10 km and lead time of $[60, 90]$ minutes.

Appendix C

Additional Results for Experiment 2

Figures C.1-C.12 are analogous to Figures 6.13-6.15 in the main body; Figures C.13-C.15 are analogous to 6.16-6.18 in the main body.

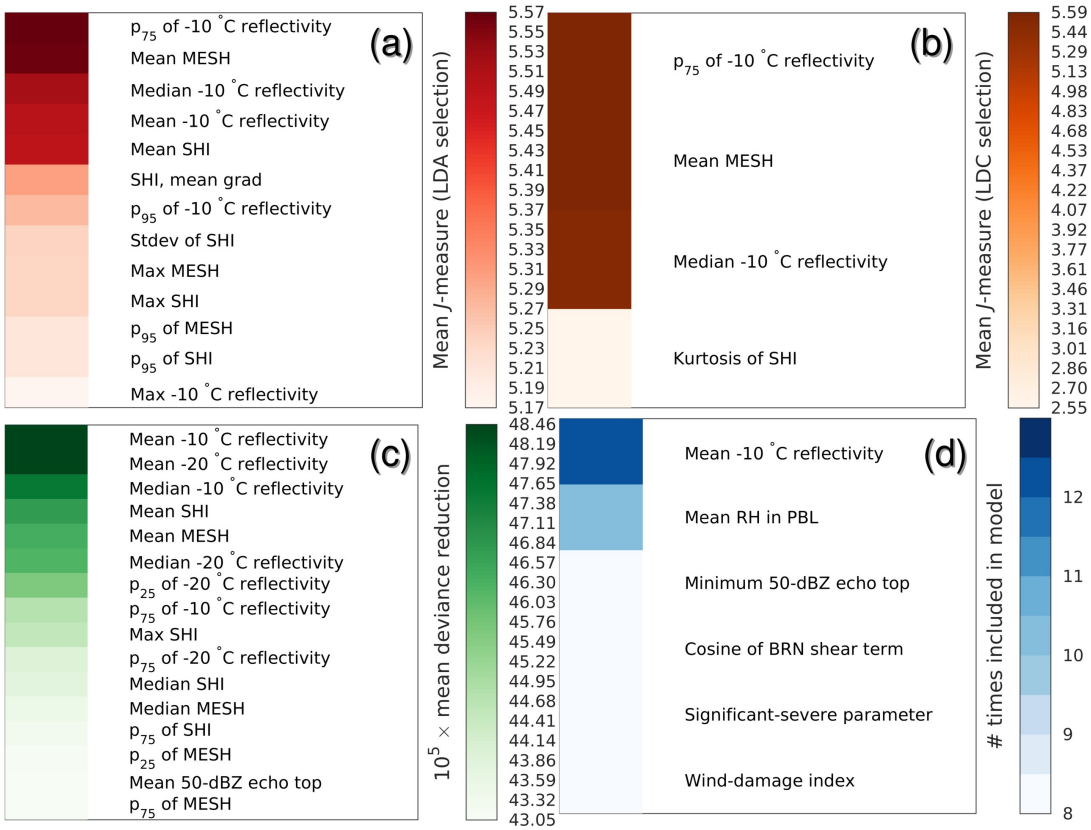

Figure C.1: Top variables for (a) J-measures with LDA selection; (b) J-measures with LDC selection; (c) decision-tree method; and (d) sequential forward selection at buffer distance of 0 km and lead time of $[15, 30]$ minutes.

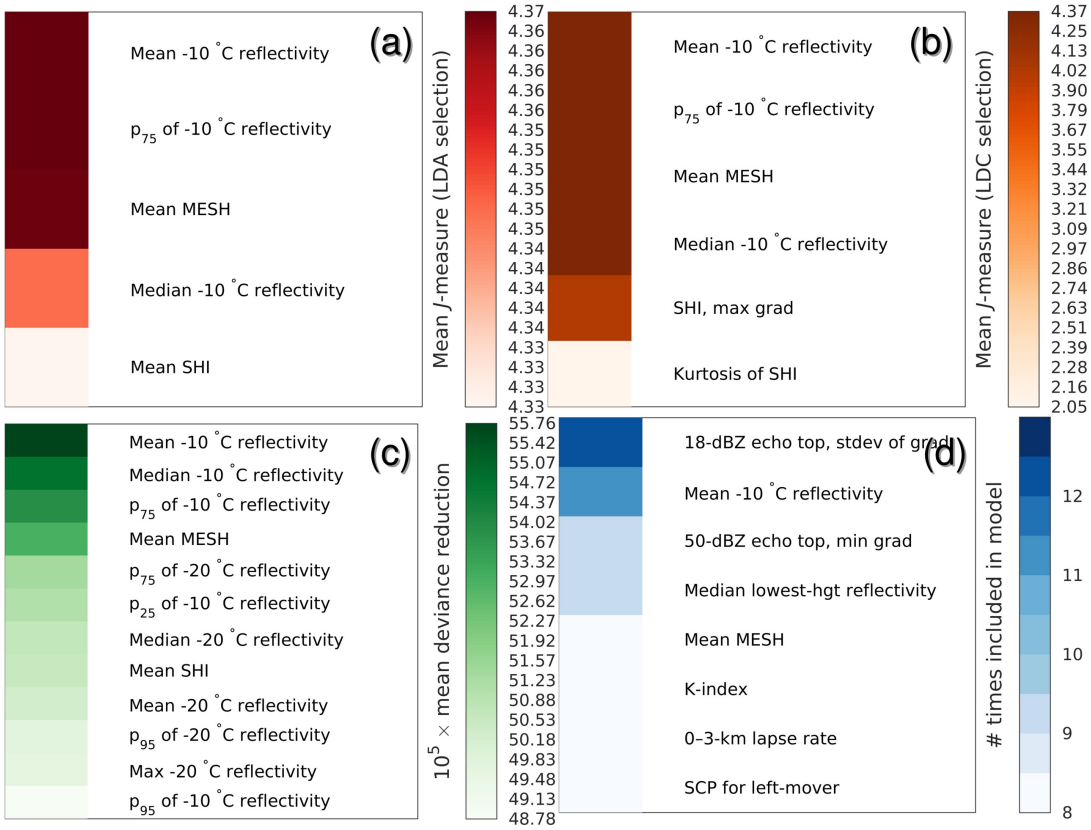

Figure C.2: As in Figure C.1, except for a buffer distance of 0 km and lead time of $[30, 45]$ minutes.

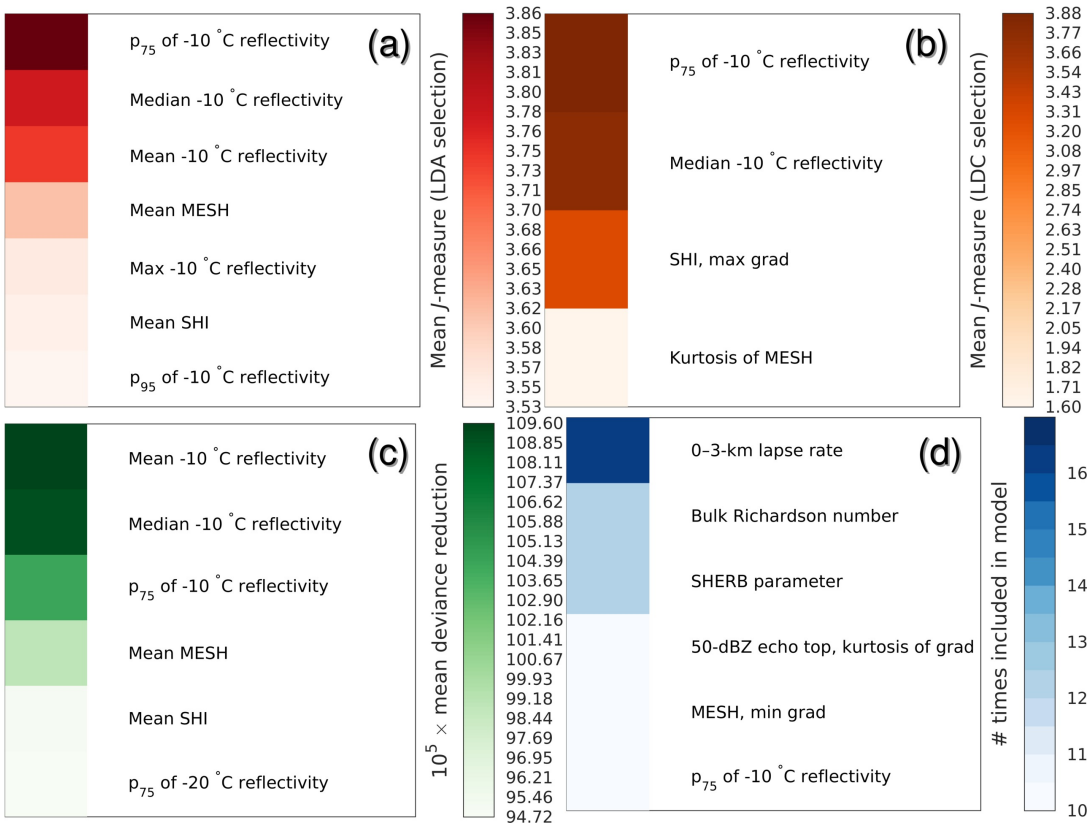

Figure C.3: As in Figure C.1, except for a buffer distance of 0 km and lead time of $[45, 60]$ minutes.

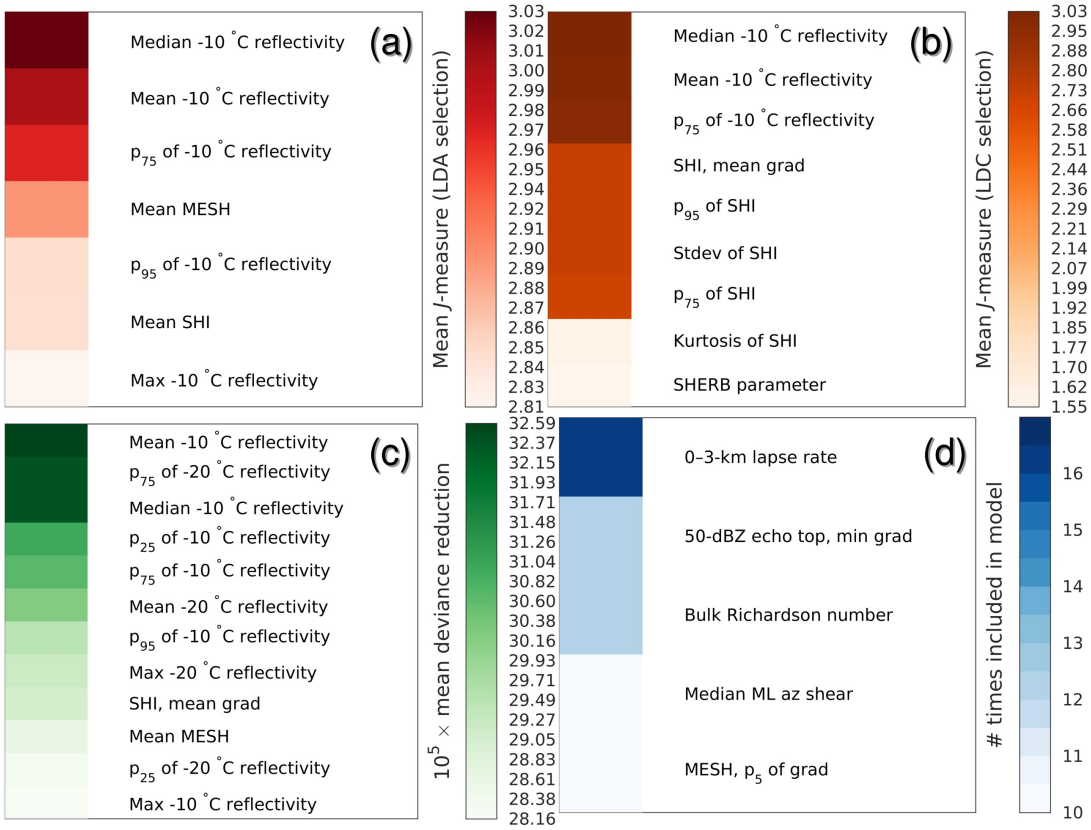

Figure C.4: As in Figure C.1, except for a buffer distance of 0 km and lead time of $[60, 90]$ minutes.

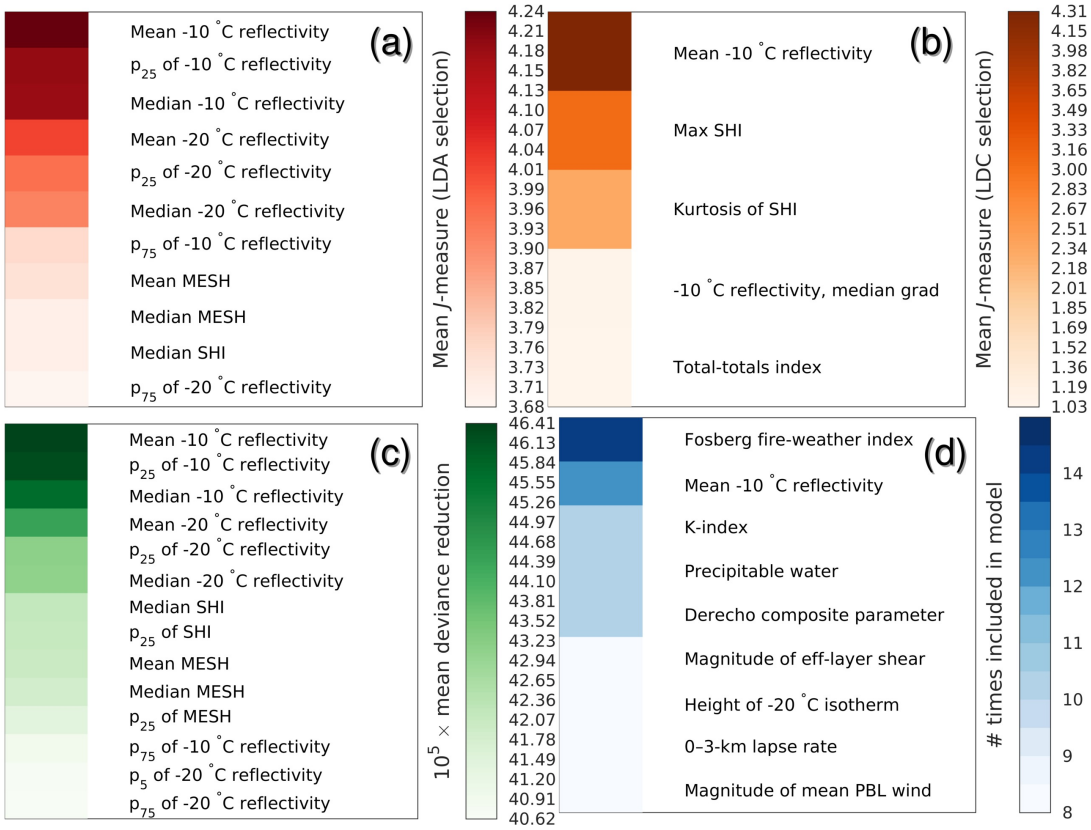

Figure C.5: As in Figure C.1, except for a buffer distance of 5 km and lead time of $[0, 15]$ minutes.

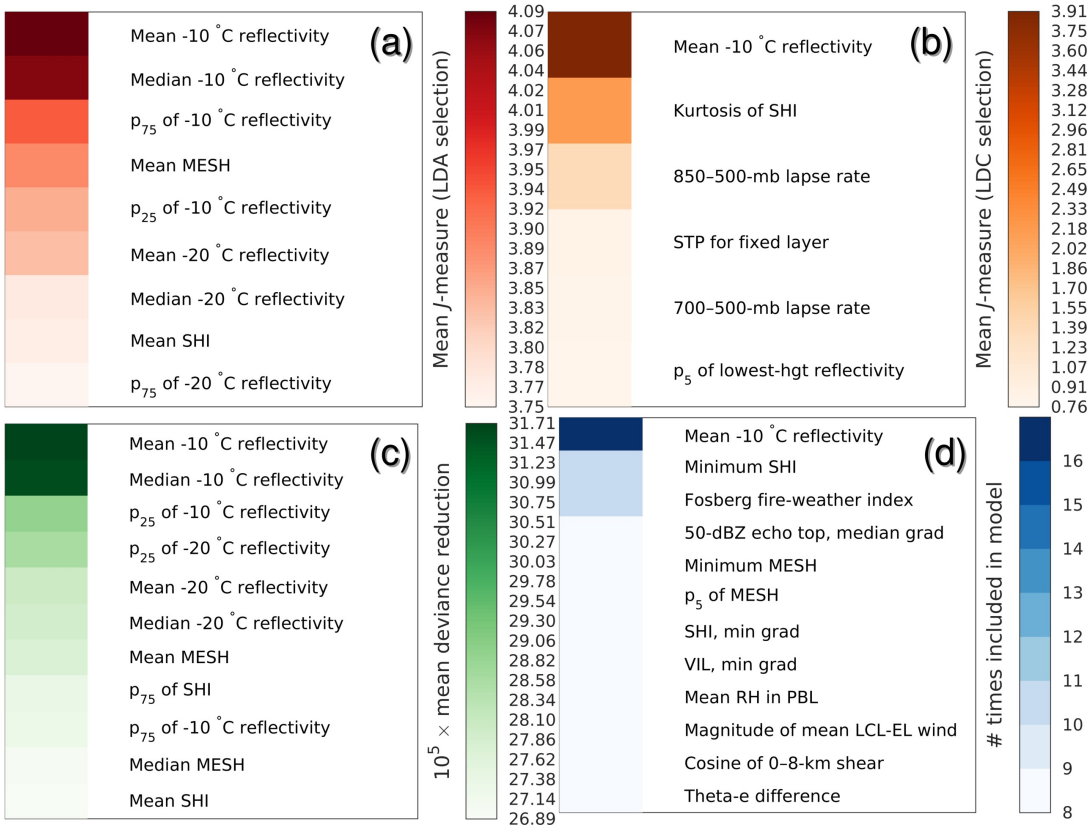

Figure C.6: As in Figure C.1, except for a buffer distance of 5 km and lead time of $[15, 30]$ minutes.

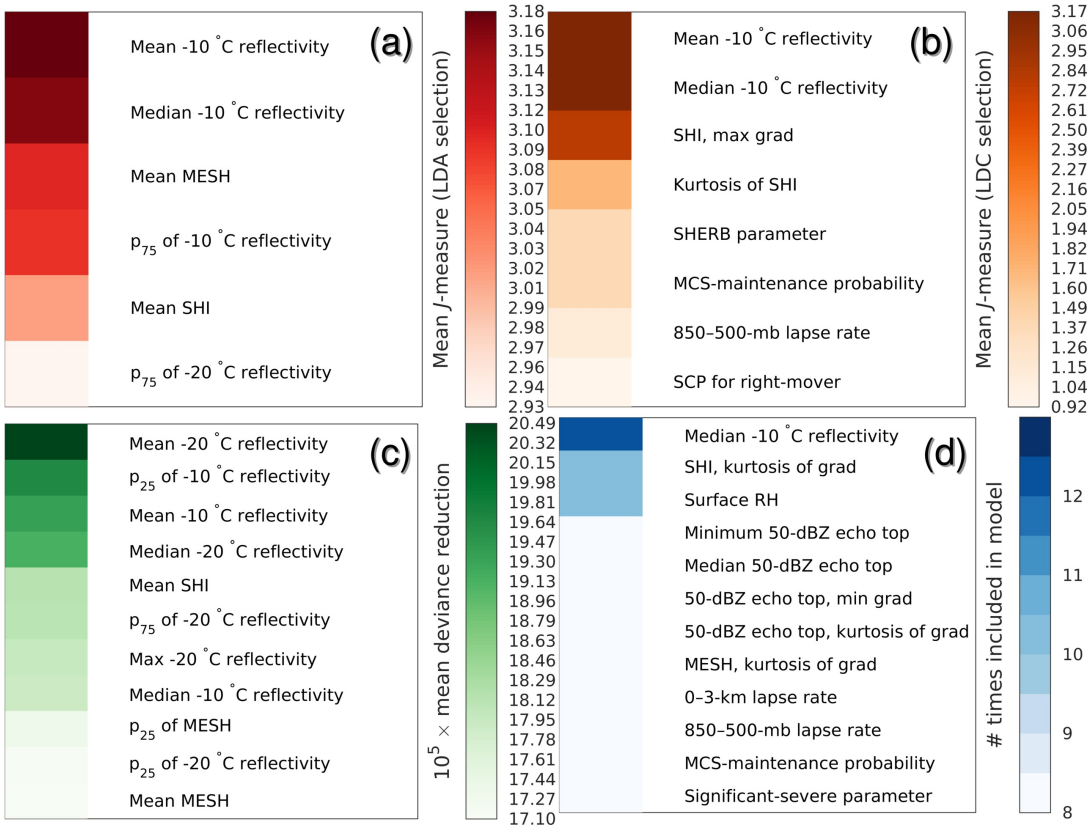

Figure C.7: As in Figure C.1, except for a buffer distance of 5 km and lead time of $[45, 60]$ minutes.

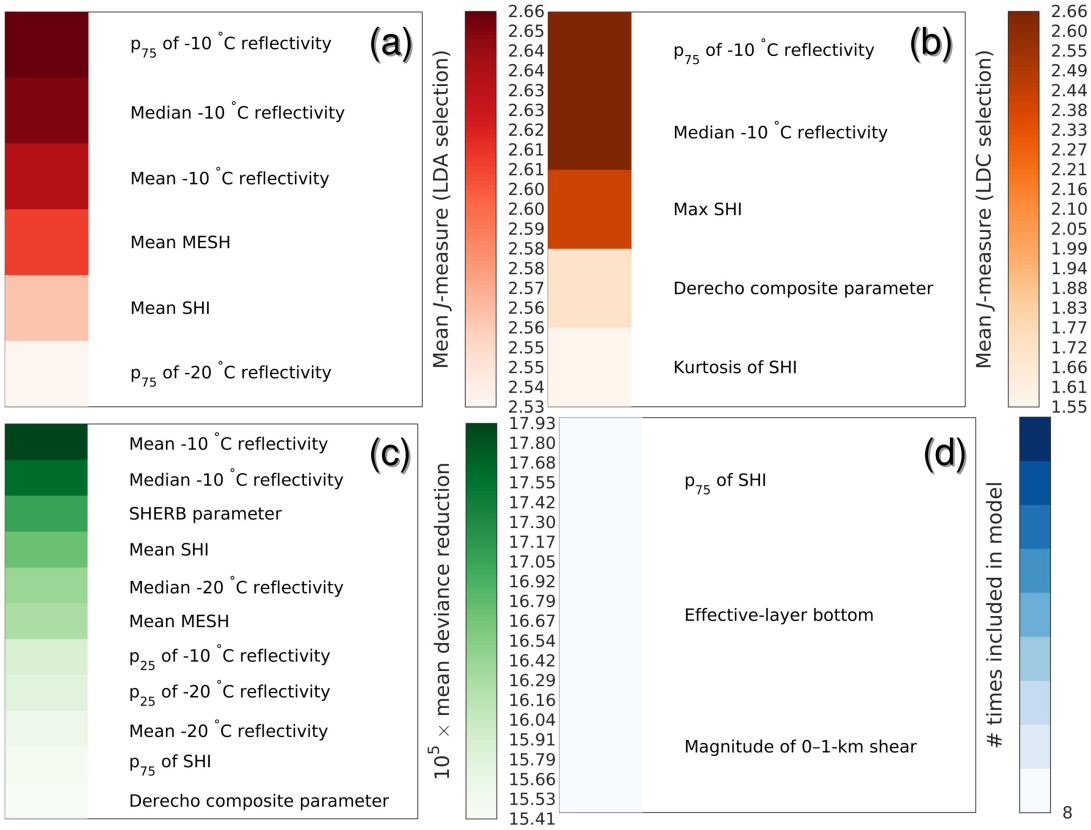

Figure C.8: As in Figure C.1, except for a buffer distance of 5 km and lead time of $[60, 90]$ minutes.

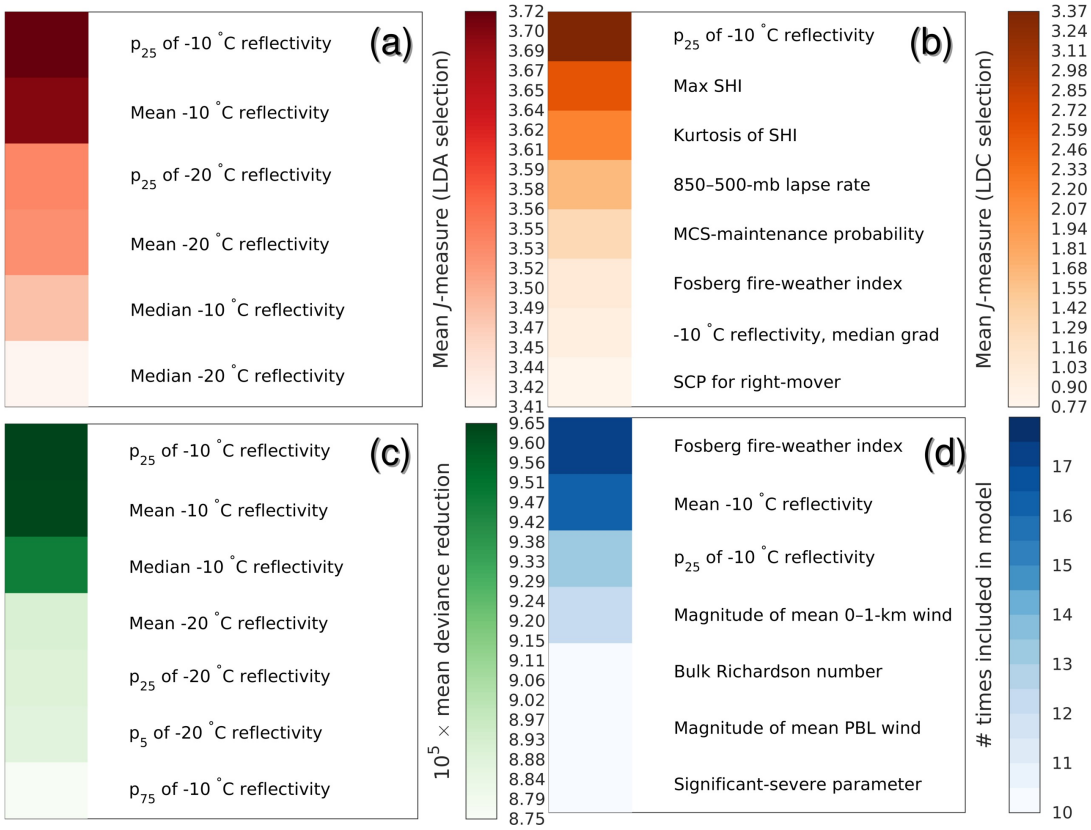

Figure C.9: As in Figure C.1, except for a buffer distance of 10 km and lead time of $[0, 15]$ minutes.

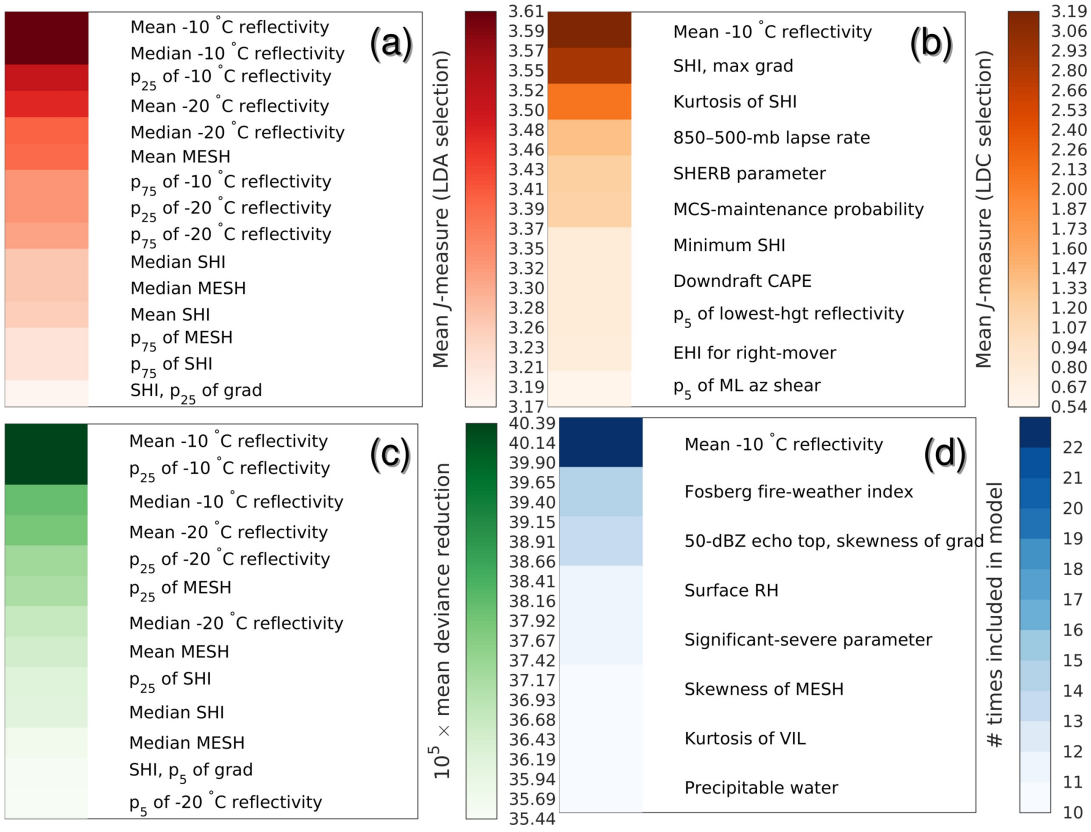

Figure C.10: As in Figure C.1, except for a buffer distance of 10 km and lead time of $[15, 30]$ minutes.

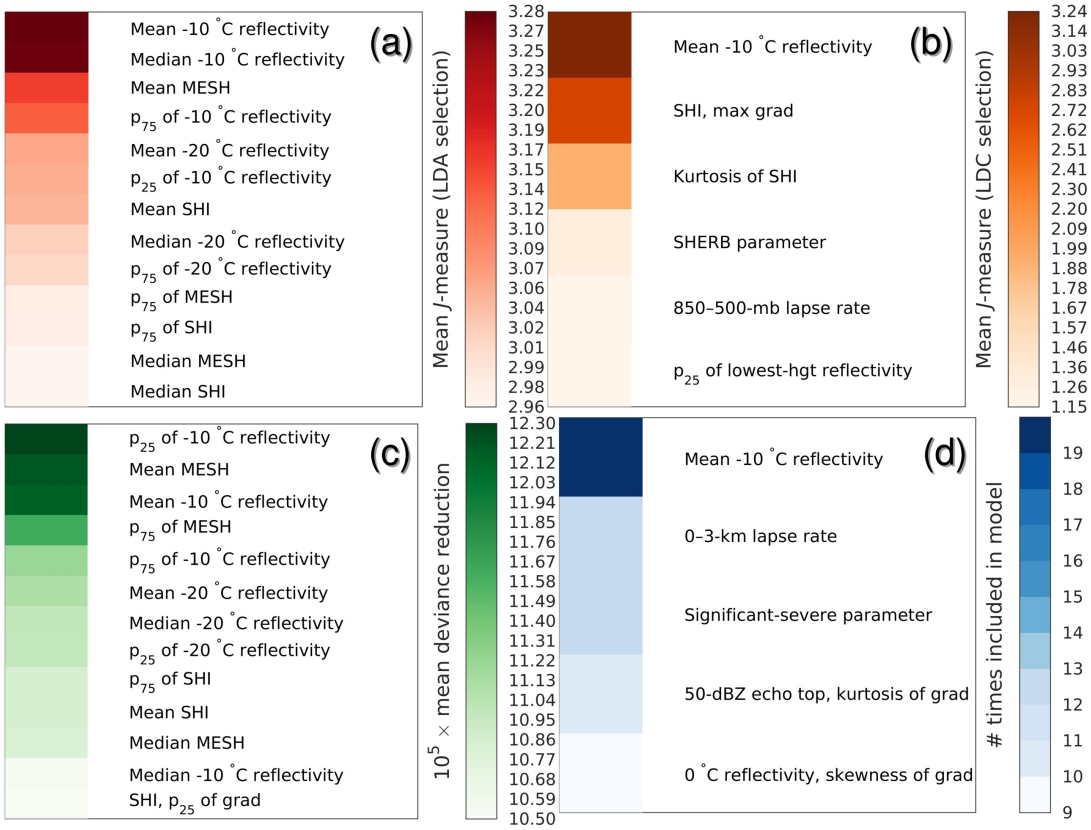

Figure C.11: As in Figure C.1, except for a buffer distance of 10 km and lead time of $[30, 45]$ minutes.

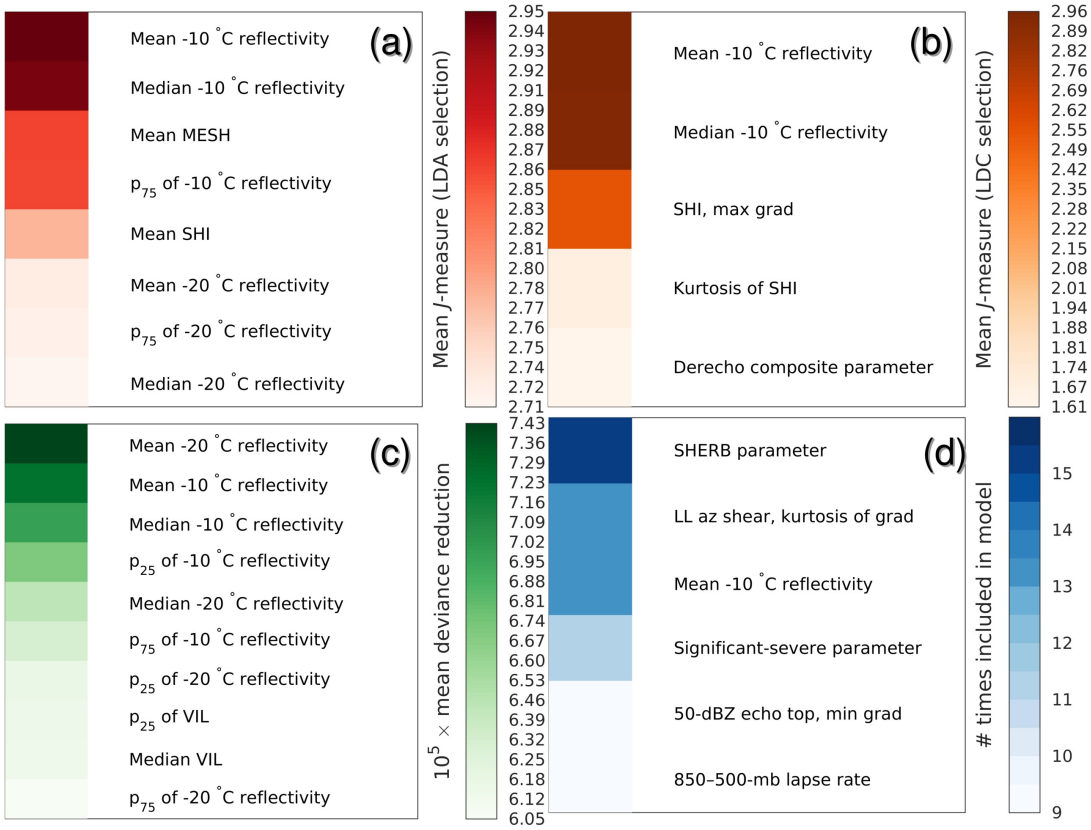

Figure C.12: As in Figure C.1, except for a buffer distance of 10 km and lead time of $[45, 60]$ minutes.

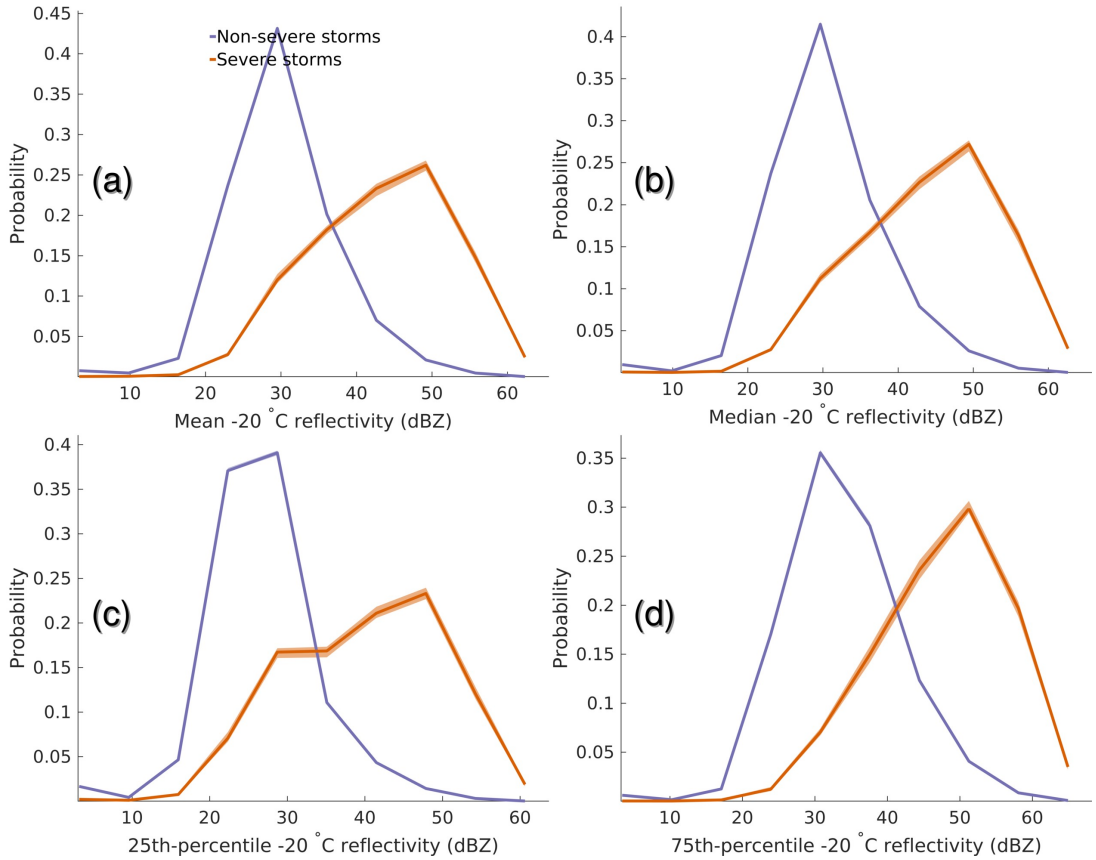

Figure C.13: Class-conditional PDFs for statistics involving -20 °C reflectivity. The "class" is the label for the storm object (1 if $U_{max} \geq 50$ kt, 0 otherwise) at a buffer distance of 5 km and lead time of $[30, 45]$ minutes. The orange curve is the PDF for storm objects with label = 1; the purple curve is for storm objects with label = 0. Each solid line is a mean, and each shaded area is a 95% confidence interval, over 25 bootstrap replicates.

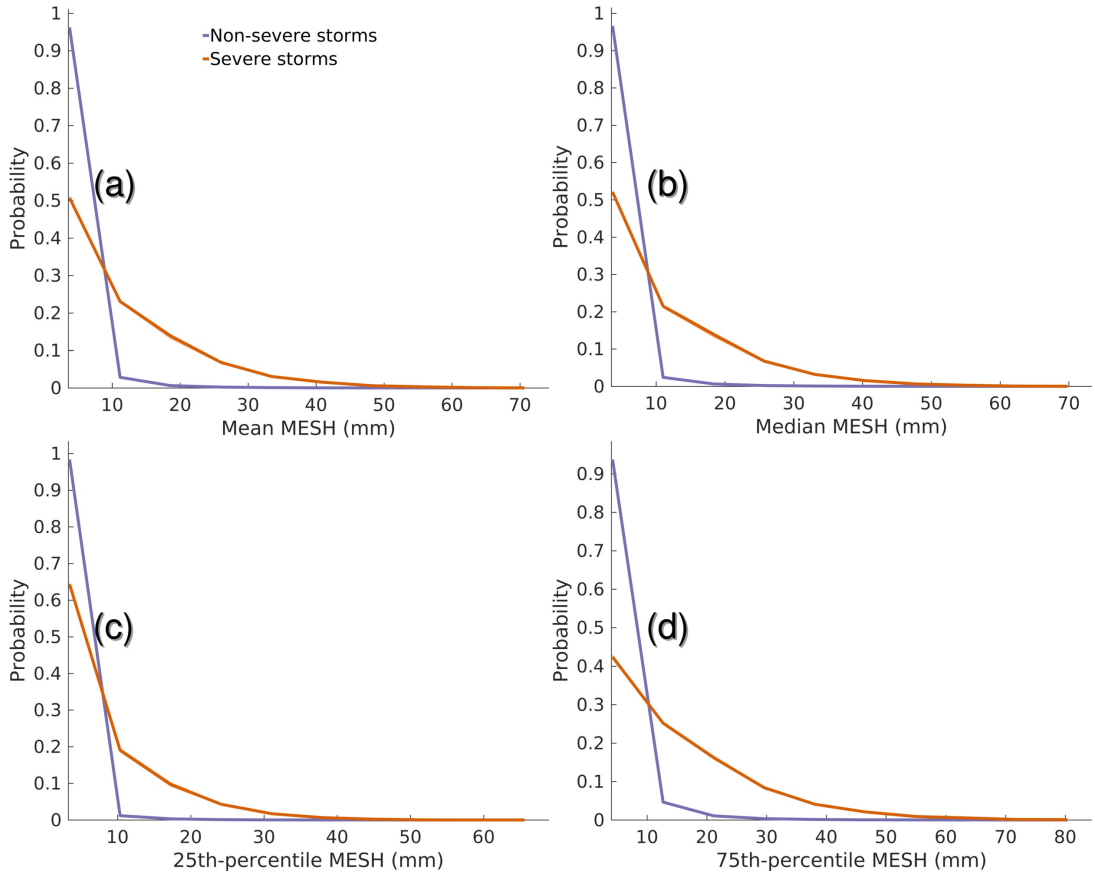

Figure C.14: Class-conditional PDFs for statistics involving maximum estimated hail size (MESH). Other details are the same as in Figure C.13.

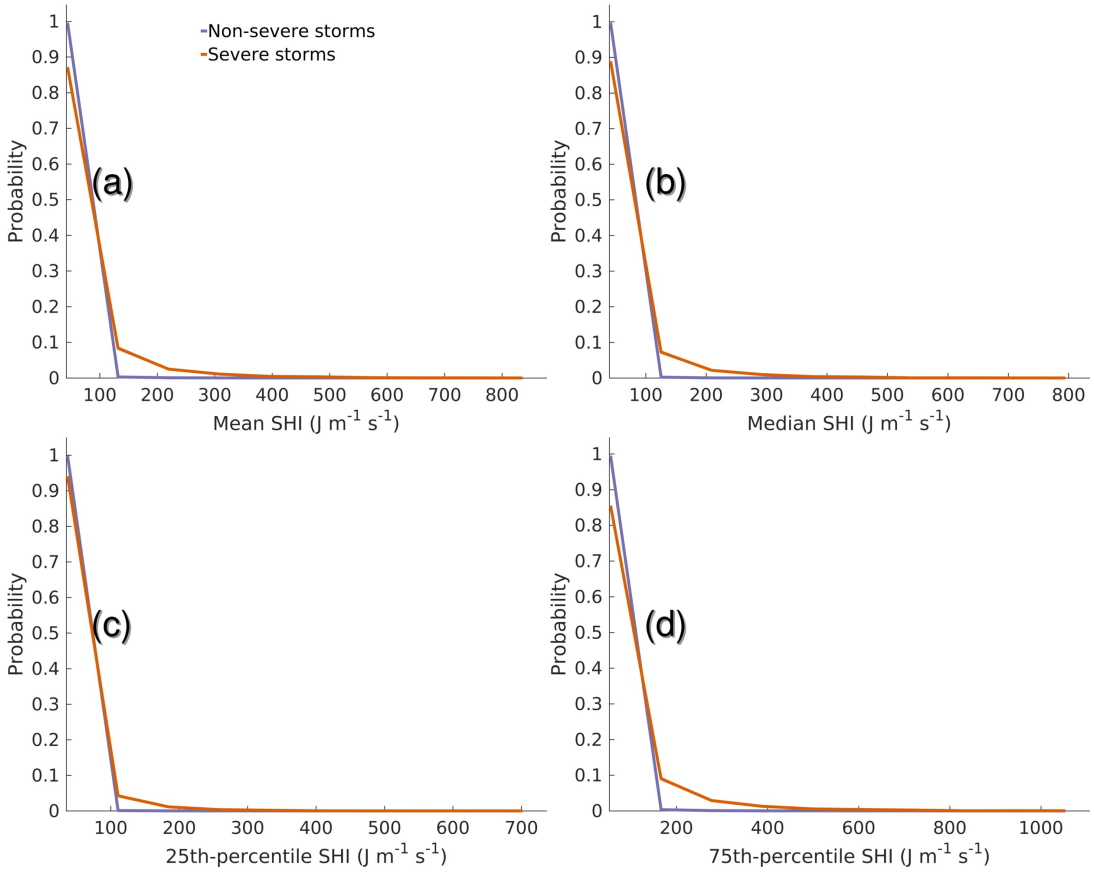

Figure C.15: Class-conditional PDFs for statistics involving severe-hail index (SHI). Other details are the same as in Figure C.13.