Supplemental Material

Weather and Forecasting
Machine Learning for Real-TimePrediction of Damaging Straight-Line Convective Wind
https://doi.org/10.1175/WAF-D-17-0038.s1

APPENDIX A

Details of Input Data

Sounding indices are defined in Table A1. The monthly distribution of training/testing days is shown Figure A1.

APPENDIX B

Experiment Details

a. Logistic Regression

Logistic regression is run with the parameters shown in Table B1, using MATLAB's `glmfit` (MathWorks 2016b). These parameters force `glmfit` to run logistic regression (`link` = `logit`) for binary classification (`distr` = `binomial`), with the bias term (β_0 in Equation 1) included (`constant` = `on`). `glmfit` is run only once for each distance buffer and time window, since there is only one choice for each parameter.

b. Logistic Regression with an Elastic Net

LREN is run with the parameters shown in Table B2, using MATLAB's `lassoglm` (MathWorks 2016c). 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 3, which determines the balance between ridge regression ($\alpha = 0$) and lasso optimization ($\alpha = 1$). Our minimum value is 0.05 rather than 0, because `lassoglm` does not accept `alpha` = 0. `numLambda` is the number of λ-values attempted (used in Equation 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 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 validation fold (Section 4b), `lassoglm` trains 50 models (one for each λ-value). We keep the model with the highest AUC on the validation fold.

For each distance buffer and time window, `lassoglm` is run 25 times (once for each of five `alpha` values and five `lambdaRatio` values).

c. Feed-forward Neural Nets

FFNN is run with the parameters shown in Table B3, using MATLAB's `patternnet` (MathWorks 2016d). 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. We kept a small number of neurons and hidden layers, because runtime increases dramatically with both and the experiment already took ~50,000 core-hours (~15 days in actual time). `trainFcn` is the backpropagation method; the two options stand for resilient backpropagation (Riedmiller and Braun 1993) and scaled conjugate gradient (SCG) (Møller 1993). `performFcn` is `crossentropy`, which is another word for deviance (Equation 2). `epochs` is the number of times that each training example is presented to the neural net. In our experience the stopping criterion (that deviance on validation examples increases between successive epochs) is 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 distance buffer and time window, `patternnet` is run 800 times (once 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`).

d. Random Forests

Random forests are run with the parameters shown in Table B4, using MATLAB's `fitensemble` (MathWorks 2016a). 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 B4 but not used for `templateTree`.

`nVarToSample` is the number of predictors to sample randomly at each split point (this sampling is called "feature-bagging" and described in Section 3a5). 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. `splitCriterion` is the objective function (Equation 2). `mergeLeaves` causes leaf nodes to be merged if they originate from the same parent node and, when applied to validation data, lead to an increase in deviance. `nLearn` is the number of decision trees in the ensemble. `resample` = `on` means that training examples are resampled for each tree (this is called "tree-bagging" and described in Section 3a5). `fResample` is the fraction of training examples resampled for each tree, and `replace` = `on` ensures that sampling is done with replacement.

For each distance buffer and time window, `fitensemble` is run 210 times to create a random forest (one for each of seven `nVarToSample` values, five `minLeaf` values, and six `fResample` values).

e. Gradient-boosted Tree Ensembles

GBEs are run with the parameters shown in Table B5, using MATLAB's `fitensemble` (MathWorks 2016a). 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 GBEs perform badly with small values. Similarly, smaller values of `fResample` are tried, because we find that GBEs perform badly with large values. `method` is the boosting method (see Freund and Schapire 1997 for AdaBoost), and `learnRate` is the learning rate for AdaBoost.

For each distance buffer and time window, `fitensemble` is run 750 times to create a GBE (one for each of five `nVarsToSample` values, five `minLeaf` values, six `fResample` values, and five `learnRate` values).

f. Isotonic Regression

Isotonic regression is run with MATLAB's `lsqisotonic` (Abonyi 2016), which requires no input parameters beyond the training data.

APPENDIX C

Experiment Results

For each distance buffer at lead times of 15-30, 30-45, and 45-60 min, Figures C1-C6 show performance of the selected model on testing data.

For all distance–lead-time pairs, parameters for the top models are shown in Tables C1-C15. "Top models" include the selected model (that with the highest validation AUC, discussed in Section 4b) and those for which validation AUC is not significantly different at the 95% level. This is determined by bootstrapping the validation set 100 times. In Tables C1-C15, "RF" is random forest and `nVar` is `nVarToSample` from Tables B4-B5. For a given distance–lead-time pair, if there are more than 20 top models, the table is truncated to 20.

References

Abonyi, 2016: Graph-based clustering and data visualization algorithms. Accessed: 2016-10-18, http://www.mathworks.com/matlabcentral/fileexchange/47196-graph-based-clustering-and-data-visualization-algorithms/content/improve_JP/toolbox_imp_JP/lsqisotonic.m.

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

MathWorks, 2016a: Fit ensemble of learners for classification and regression – MATLAB fitensemble. Accessed: 2016-10-18, http://www.mathworks.com/help/stats/fitensemble.html.

MathWorks, 2016b: Generalized linear model regression – MATLAB glmfit. Accessed: 2016-10-18, http://www.mathworks.com/help/stats/glmfit.html.

MathWorks, 2016c: Lasso or elastic net regularization for generalized linear model regression – MATLAB lassoglm. Accessed: 2016-10-18, http://www.mathworks.com/help/stats/lassoglm.html.

MathWorks, 2016d: Pattern Recognition Network – MATLAB patternnet. Accessed: 2016-09-24, http://www.mathworks.com/help/nnet/ref/patternnet.html.

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

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.

List of Tables

Table A1: Sounding indices (part 1 of 5).

Variable	Description	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	
`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	$\circ$	
`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}	

Table A1: Sounding indices (part 2 of 5).

Variable	Description	Units	Vector?
`ehi3km`	EHI from 0-3 km AGL	J kg^{-1}	
`ehiLeft`	Effective-layer EHI for left-mover	J kg^{-1}	
`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	
`liMax`	Max lifted index in column (surface – any level)	K	

Table A1: Sounding indices (part 3 of 5).

Variable	Description	Units	Vector?
`maxWindPbl`	Max wind in planetary boundary layer (PBL)	m s^{-1}	Yes
`mcp`	Microburst composite parameter	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	
`mmp`	Mesoscale convective system (MCS) maintenance probability	%	
`mplHeight`	Height AGL of max parcel level	m	
`pblDepth`	Depth of PBL	m	

Table A1: Sounding indices (part 4 of 5).

Variable	Description	Units	Vector?
`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}	
`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

Table A1: Sounding indices (part 5 of 5).

Variable	Description	Units	Vector?
`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	
`thetaeIndex`	θ_e-index	K	
`totalTotals`	Total-totals index	K	
`updraftTilt`	Updraft tilt	$^\circ$	
`verticalTotals`	Vertical-totals index	K	
`wdp`	Wind-damage parameter	None	

Table B1: Parameters for logistic regression, using MATLAB's `glmfit`.

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

Table B2: Parameters for LREN, using MATLAB's `lassoglm`.

Parameter	Values
`distr`	`binomial`
`link`	`logit`
`alpha`	0.05, 0.25, 0.5, 0.75, 1
`numLambda`	50
`lambdaRatio`	10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}
`relTol`	0.01

Table B3: Parameters for FFNN, using MATLAB's `patternnet`. For parameters with more than one option, any value in bold occurred for one of the 15 selected models.

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`	$\mathbf{5\times10^{-7}}$, $\mathbf{5\times10^{-6}}$, 5×10^{-5}, 5×10^{-4}, $\mathbf{5\times10^{-3}}$
`lambda`	5×10^{-9}, 5×10^{-8}, $\mathbf{5\times10^{-7}}$, $\mathbf{5\times10^{-6}}$, $\mathbf{5\times10^{-5}}$

Table B4: Parameters for random forests, using MATLAB's `fitensemble`. For parameters with more than one option, any value in bold occurred for one of the 15 selected models.

Parameter	**Values**
`nVarToSample`	1, 5, 10, 25, 50, 75, **100**
`minLeaf`	5, 10, **25**, 50, 75
`splitCriterion`	`deviance`
`mergeLeaves`	`on`
`nLearn`	`250`
`resample`	`on`
`fResample`	0.05, 0.20, 0.40, 0.60, 0.80, **1.00**
`replace`	`on`

Table B5: Parameters for GBEs, using MATLAB's `fitensemble`. For parameters with more than one option, any value in bold occurred for one of the 15 selected models.

Parameter	Values
`nVarToSample`	**10**, 25, **50**, **100**, **431**
`minLeaf`	**5**, 10, **25**, **50**, **75**
`splitCriterion`	`deviance`
`mergeLeaves`	`on`
`method`	`AdaBoostM1`
`nLearn`	`250`
`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 C1: Top models for 0 km (inside storm) and 0–15-minute lead time. First row is selected model; subsequent rows are models for which validation AUC is not significantly different at the 95% level.

Model	**AUC**	**Parameters**
GBE	0.9986	`fResample` = 0.20; `nVar` = 100; `minLeaf` = 50; `learnRate` = 0.20
GBE	0.9874	`fResample` = 0.20; `nVar` = 10; `minLeaf` = 25; `learnRate` = 0.05

Table C2: As in Table C1 but for 15–30-minute lead time.

Model	**AUC**	**Parameters**
FFNN	0.9625	$\{20, 10\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-7}$; `lambda` $= 5 \times 10^{-6}$
LREN	0.9616	`alpha` $= 0.05$, `lambdaRatio` $= 10^{-6}$
LREN	0.9592	`alpha` $= 0.50$, `lambdaRatio` $= 10^{-5}$
GBE	0.9586	`fResample` $= 0.10$; `nVar` $= 431$; `minLeaf` $= 25$; `learnRate` $= 0.05$
GBE	0.9577	`fResample` $= 0.15$; `nVar` $= 50$; `minLeaf` $= 10$; `learnRate` $= 0.05$
LREN	0.9577	`alpha` $= 0.75$, `lambdaRatio` $= 10^{-6}$
RF	0.9574	`fResample` $= 0.60$; `nVarsToSample` $= 25$; `minLeaf` $= 25$
GBE	0.9574	`fResample` $= 0.10$; `nVar` $= 10$; `minLeaf` $= 5$; `learnRate` $= 0.10$
GBE	0.9566	`fResample` $= 0.15$; `nVar` $= 10$; `minLeaf` $= 50$; `learnRate` $= 0.10$
LREN	0.9565	`alpha` $= 0.50$, `lambdaRatio` $= 10^{-3}$
GBE	0.9549	`fResample` $= 0.10$; `nVar` $= 50$; `minLeaf` $= 10$; `learnRate` $= 0.15$
RF	0.9545	`fResample` $= 0.20$; `nVarsToSample` $= 100$; `minLeaf` $= 10$
RF	0.9544	`fResample` $= 0.40$; `nVarsToSample` $= 50$; `minLeaf` $= 25$
GBE	0.9542	`fResample` $= 0.10$; `nVar` $= 100$; `minLeaf` $= 25$; `learnRate` $= 0.05$
RF	0.9535	`fResample` $= 0.80$; `nVarsToSample` $= 25$; `minLeaf` $= 5$
GBE	0.9532	`fResample` $= 0.15$; `nVar` $= 10$; `minLeaf` $= 10$; `learnRate` $= 0.10$
RF	0.9525	`fResample` $= 0.60$; `nVarsToSample` $= 100$; `minLeaf` $= 5$
RF	0.9500	`fResample` $= 0.05$; `nVarsToSample` $= 50$; `minLeaf` $= 10$
GBE	0.9486	`fResample` $= 0.10$; `nVar` $= 10$; `minLeaf` $= 5$; `learnRate` $= 0.01$
RF	0.9481	`fResample` $= 0.20$; `nVarsToSample` $= 100$; `minLeaf` $= 5$

Table C3: As in Table C1 but for 30–45-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9486	`fResample` = 0.05; `nVar` = 100; `minLeaf` = 25; `learnRate` = 0.15
FFNN	0.9436	$\{5, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-6}

Table C4: As in Table C4 but for 45–60-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9354	`fResample` = 0.20; `nVar` = 431; `minLeaf` = 75; `learnRate` = 0.10

Table C5: As in Table C5 but for 60–90-minute lead time.

Model	**AUC**	**Parameters**
FFNN	0.9385	$\{15, 15\}$ neurons; `trainrp`; `delt_dec` $= 0.900$; `delt_inc` $= 1.1$
GBE	0.9344	`fResample` $= 0.05$; `nVar` $= 50$; `minLeaf` $= 5$; `learnRate` $= 0.01$

Table C6: As in Table C1 but for 0-5 km outside storm and 0–15-minute lead time.

Model	AUC	Parameters
RF	0.9677	`fResample` = 1.00; `nVarsToSample` = 100; `minLeaf` = 25

Table C7: As in Table C1 but for 0-5 km outside storm and 15–30-minute lead time.

Model	**AUC**	**Parameters**
FFNN	0.9515	$\{20, 15\}$ neurons; `trainscg`; `sigma` = 5×10^{-6}; `lambda` = 5×10^{-6}
GBE	0.9498	`fResample` = 0.05; `nVar` = 10; `minLeaf` = 75; `learnRate` = 0.15
GBE	0.9496	`fResample` = 0.20; `nVar` = 25; `minLeaf` = 10; `learnRate` = 0.01
GBE	0.9490	`fResample` = 0.20; `nVar` = 10; `minLeaf` = 75; `learnRate` = 0.10
GBE	0.9484	`fResample` = 0.25; `nVar` = 25; `minLeaf` = 25; `learnRate` = 0.05
GBE	0.9483	`fResample` = 0.25; `nVar` = 10; `minLeaf` = 5; `learnRate` = 0.10
GBE	0.9481	`fResample` = 0.30; `nVar` = 25; `minLeaf` = 10; `learnRate` = 0.10
RF	0.9478	`fResample` = 0.60; `nVarsToSample` = 25; `minLeaf` = 5
GBE	0.9471	`fResample` = 0.30; `nVar` = 25; `minLeaf` = 50; `learnRate` = 0.05
GBE	0.9469	`fResample` = 0.10; `nVar` = 431; `minLeaf` = 10; `learnRate` = 0.10
GBE	0.9469	`fResample` = 0.15; `nVar` = 10; `minLeaf` = 75; `learnRate` = 0.05
GBE	0.9466	`fResample` = 0.25; `nVar` = 50; `minLeaf` = 5; `learnRate` = 0.10
RF	0.9466	`fResample` = 0.40; `nVarsToSample` = 100; `minLeaf` = 25
GBE	0.9453	`fResample` = 0.20; `nVar` = 50; `minLeaf` = 5; `learnRate` = 0.05
RF	0.9452	`fResample` = 1.00; `nVarsToSample` = 75; `minLeaf` = 10
GBE	0.9449	`fResample` = 0.10; `nVar` = 100; `minLeaf` = 75; `learnRate` = 0.15
GBE	0.9440	`fResample` = 0.20; `nVar` = 431; `minLeaf` = 5; `learnRate` = 0.10
GBE	0.9435	`fResample` = 0.25; `nVar` = 25; `minLeaf` = 5; `learnRate` = 0.01
RF	0.9432	`fResample` = 0.40; `nVarsToSample` = 25; `minLeaf` = 25
GBE	0.9405	`fResample` = 0.20; `nVar` = 50; `minLeaf` = 75; `learnRate` = 0.05

Table C8: As in Table C1 but for 0-5 km outside storm and 30–45-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9394	`fResample` = 0.20; `nVar` = 50; `minLeaf` = 5; `learnRate` = 0.10
FFNN	0.9384	$\{15, 10\}$ neurons; `trainscg`; `sigma` = 5×10^{-4}; `lambda` = 5×10^{-8}
GBE	0.9318	`fResample` = 0.15; `nVar` = 50; `minLeaf` = 10; `learnRate` = 0.10

Table C9: As in Table C1 but for 0-5 km outside storm and 45–60-minute lead time.

Model	**AUC**	**Parameters**
FFNN	0.9326	$\{15, 5\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-7}$; `lambda` $= 5 \times 10^{-7}$
GBE	0.9315	`fResample` $= 0.25$; `nVar` $= 25$; `minLeaf` $= 5$; `learnRate` $= 0.15$

Table C10: As in Table C1 but for 0-5 km outside storm and 60–90-minute lead time.

Model	AUC	Parameters
GBE	0.9304	fResample = 0.10; nVar = 50; minLeaf = 5; learnRate = 0.01
LREN	0.9298	alpha = 0.75, lambdaRatio = 10^{-3}
FFNN	0.9279	$\{20, 20\}$ neurons; trainscg; sigma = 5×10^{-7}; lambda = 5×10^{-6}
GBE	0.9266	fResample = 0.10; nVar = 10; minLeaf = 10; learnRate = 0.05
GBE	0.9260	fResample = 0.20; nVar = 50; minLeaf = 10; learnRate = 0.05
GBE	0.9221	fResample = 0.25; nVar = 50; minLeaf = 75; learnRate = 0.10

Table C11: As in Table C1 but for 5-10 km outside storm and 0–15-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9617	`fResample` = 0.15; `nVar` = 431; `minLeaf` = 75; `learnRate` = 0.10
RF	0.9611	`fResample` = 0.40; `nVarsToSample` = 50; `minLeaf` = 10
GBE	0.9610	`fResample` = 0.10; `nVar` = 100; `minLeaf` = 10; `learnRate` = 0.05
FFNN	0.9531	$\{20, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-5}
GBE	0.9519	`fResample` = 0.05; `nVar` = 100; `minLeaf` = 75; `learnRate` = 0.10

Table C12: As in Table C1 but for 5-10 km outside storm and 15–30-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9539	fResample = 0.30; nVar = 100; minLeaf = 25; learnRate = 0.10
GBE	0.9520	fResample = 0.10; nVar = 50; minLeaf = 25; learnRate = 0.05
GBE	0.9515	fResample = 0.20; nVar = 100; minLeaf = 75; learnRate = 0.05
GBE	0.9515	fResample = 0.25; nVar = 10; minLeaf = 25; learnRate = 0.05
GBE	0.9504	fResample = 0.30; nVar = 10; minLeaf = 75; learnRate = 0.05
GBE	0.9484	fResample = 0.25; nVar = 431; minLeaf = 5; learnRate = 0.10
GBE	0.9482	fResample = 0.20; nVar = 431; minLeaf = 25; learnRate = 0.05
GBE	0.9475	fResample = 0.25; nVar = 100; minLeaf = 5; learnRate = 0.10
RF	0.9446	fResample = 0.20; nVarsToSample = 5; minLeaf = 5
FFNN	0.9441	$\{10, 20\}$ neurons; trainscg; sigma = 5×10^{-5}; lambda = 5×10^{-9}
FFNN	0.9428	$\{10, 5\}$ neurons; trainscg; sigma = 5×10^{-7}; lambda = 5×10^{-6}
RF	0.9427	fResample = 0.60; nVarsToSample = 75; minLeaf = 25
RF	0.9379	fResample = 0.05; nVarsToSample = 75; minLeaf = 5

Table C13: As in Table C1 but for 5-10 km outside storm and 30–45-minute lead time.

Model	AUC	Parameters
FFNN	0.9344	$\{5, 5\}$ neurons; `trainscg`; `sigma` = 5×10^{-3}; `lambda` = 5×10^{-7}
RF	0.9318	`fResample` = 1.00; `nVarsToSample` = 5; `minLeaf` = 5
RF	0.9309	`fResample` = 0.20; `nVarsToSample` = 50; `minLeaf` = 25
RF	0.9307	`fResample` = 0.80; `nVarsToSample` = 50; `minLeaf` = 25
GBE	0.9307	`fResample` = 0.05; `nVar` = 100; `minLeaf` = 10; `learnRate` = 0.15
LREN	0.9275	`alpha` = 0.50, `lambdaRatio` = 10^{-5}
RF	0.9273	`fResample` = 0.20; `nVarsToSample` = 10; `minLeaf` = 25
RF	0.9268	`fResample` = 0.05; `nVarsToSample` = 100; `minLeaf` = 10
RF	0.9250	`fResample` = 0.60; `nVarsToSample` = 100; `minLeaf` = 75
GBE	0.9226	`fResample` = 0.05; `nVar` = 10; `minLeaf` = 75; `learnRate` = 0.20
RF	0.9223	`fResample` = 0.40; `nVarsToSample` = 25; `minLeaf` = 25
RF	0.9216	`fResample` = 1.00; `nVarsToSample` = 75; `minLeaf` = 50
GBE	0.9215	`fResample` = 0.05; `nVar` = 25; `minLeaf` = 75; `learnRate` = 0.10
RF	0.9212	`fResample` = 1.00; `nVarsToSample` = 5; `minLeaf` = 25
RF	0.9208	`fResample` = 0.80; `nVarsToSample` = 100; `minLeaf` = 10
GBE	0.9189	`fResample` = 0.05; `nVar` = 431; `minLeaf` = 10; `learnRate` = 0.01
RF	0.9182	`fResample` = 1.00; `nVarsToSample` = 1; `minLeaf` = 50
RF	0.9175	`fResample` = 0.20; `nVarsToSample` = 75; `minLeaf` = 5
LREN	0.9174	`alpha` = 0.75, `lambdaRatio` = 10^{-5}
GBE	0.9166	`fResample` = 0.05; `nVar` = 10; `minLeaf` = 5; `learnRate` = 0.01

Table C14: As in Table C1 but for 5-10 km outside storm and 45–60-minute lead time.

Model	**AUC**	**Parameters**
FFNN	0.9454	$\{15, 15\}$ neurons; `trainscg`; `sigma` $= 5 \times 10^{-7}$; `lambda` $= 5 \times 10^{-5}$

Table C15: As in Table C1 but for 5-10 km outside storm and 60–90-minute lead time.

Model	**AUC**	**Parameters**
GBE	0.9325	fResample = 0.10; nVar = 10; minLeaf = 25; learnRate = 0.05
GBE	0.9295	fResample = 0.25; nVar = 10; minLeaf = 75; learnRate = 0.15
GBE	0.9281	fResample = 0.10; nVar = 100; minLeaf = 10; learnRate = 0.20
GBE	0.9249	fResample = 0.15; nVar = 25; minLeaf = 25; learnRate = 0.15
GBE	0.9236	fResample = 0.25; nVar = 25; minLeaf = 75; learnRate = 0.10
GBE	0.9227	fResample = 0.15; nVar = 100; minLeaf = 5; learnRate = 0.10
FFNN	0.9218	$\{15, 10\}$ neurons; trainscg; sigma = 5×10^{-7}; lambda = 5×10^{-5}
FFNN	0.9202	$\{10, 20\}$ neurons; trainscg; sigma = 5×10^{-5}; lambda = 5×10^{-8}
GBE	0.9195	fResample = 0.15; nVar = 431; minLeaf = 5; learnRate = 0.01
GBE	0.9193	fResample = 0.15; nVar = 10; minLeaf = 10; learnRate = 0.05
RF	0.9188	fResample = 0.80; nVarsToSample = 75; minLeaf = 25
GBE	0.9168	fResample = 0.15; nVar = 10; minLeaf = 50; learnRate = 0.05
RF	0.9103	fResample = 0.40; nVarsToSample = 100; minLeaf = 25
FFNN	0.9064	$\{15, 20\}$ neurons; trainrp; delt_dec = 0.500; delt_inc = 1.1
GBE	0.9047	fResample = 0.15; nVar = 431; minLeaf = 75; learnRate = 0.15

List of Figures

Fig. A1: Monthly distribution of training/testing days.

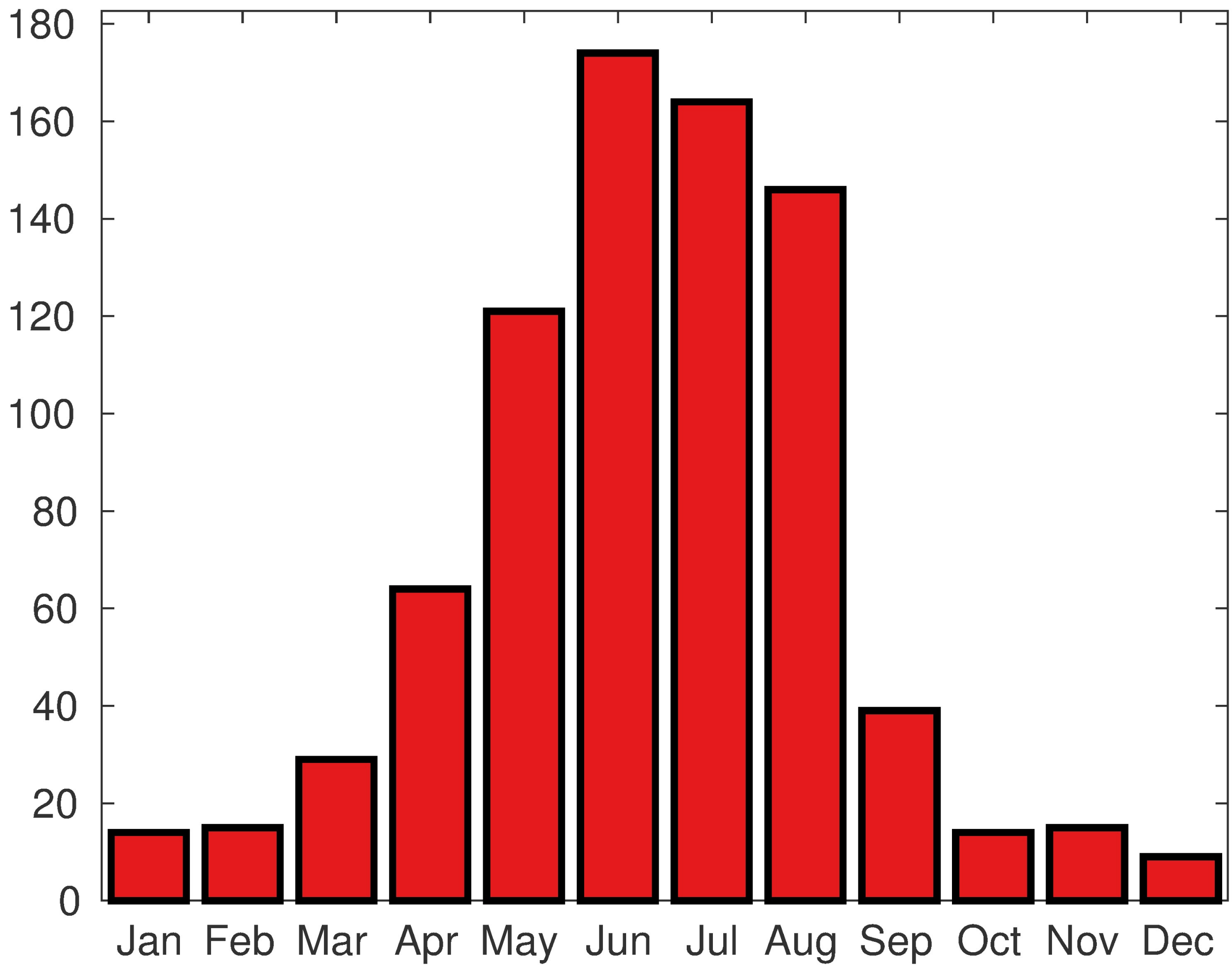

Fig. C1: As in Figure 9 but for 15–30-min lead time. Selected models (from nearest to farthest distance buffer) are an FFNN, FFNN, and GBE.

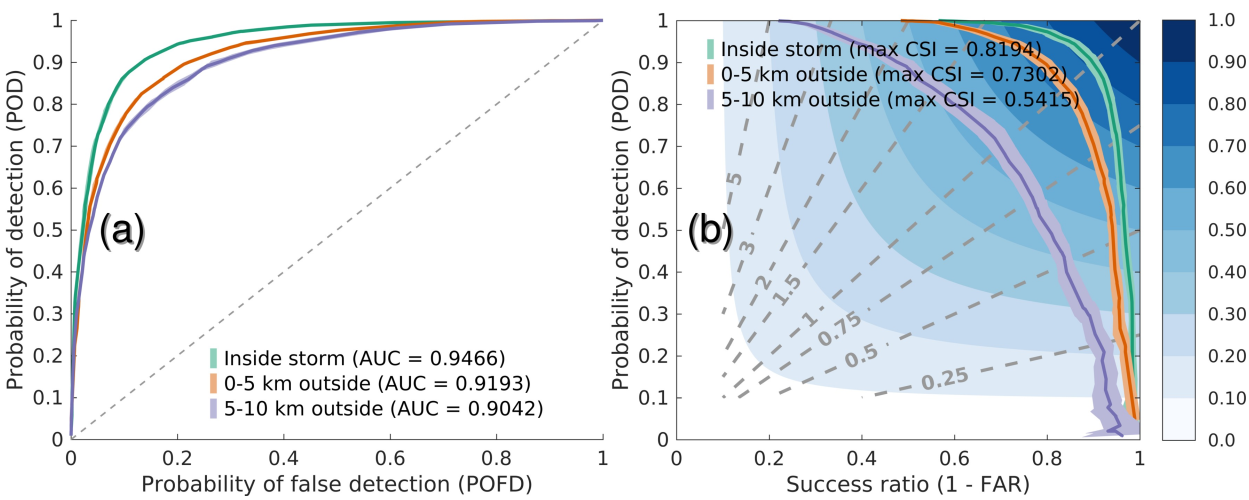

Fig. C2: As in Figure 9 but for 30–45-min lead time. Selected models (from nearest to farthest distance buffer) are a GBE, GBE, and FFNN.

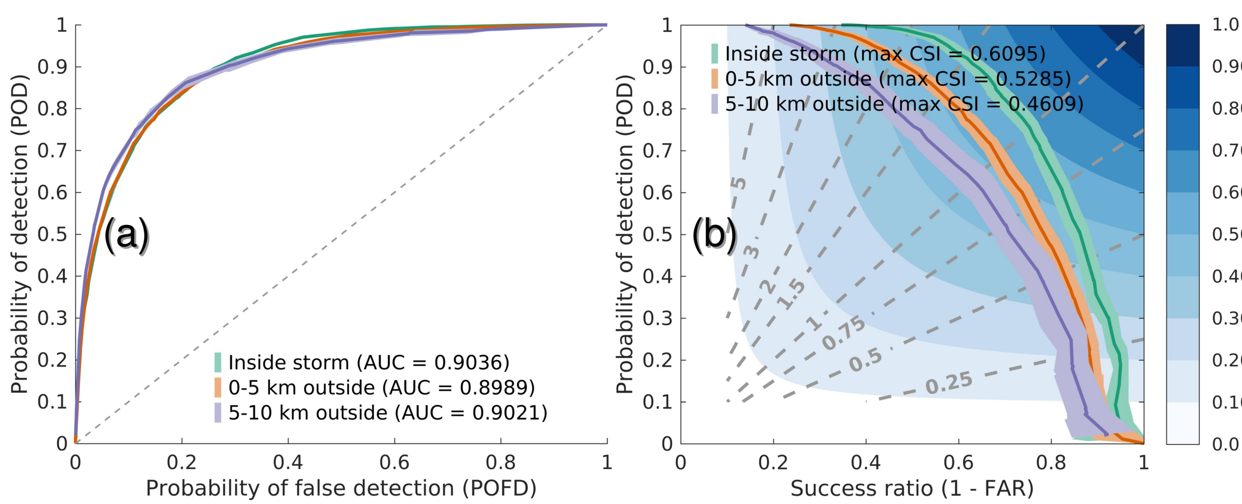

Fig. C3: As in Figure 9 but for 45–60-min lead time. Selected models (from nearest to farthest distance buffer) are a GBE, FFNN, and FFNN.

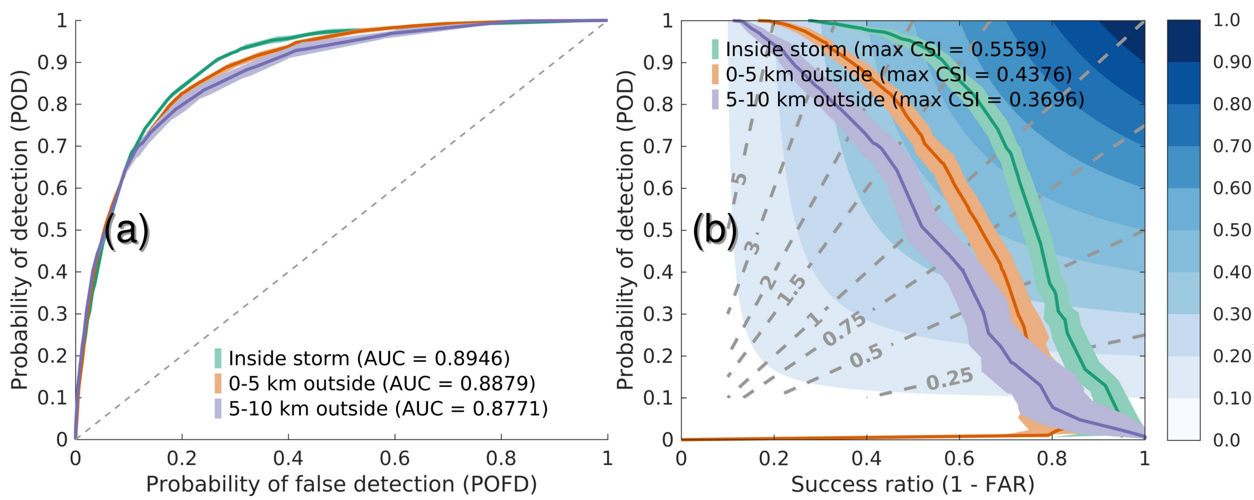

Fig. C4: As in Figure 10 but for 15–30-min lead time.

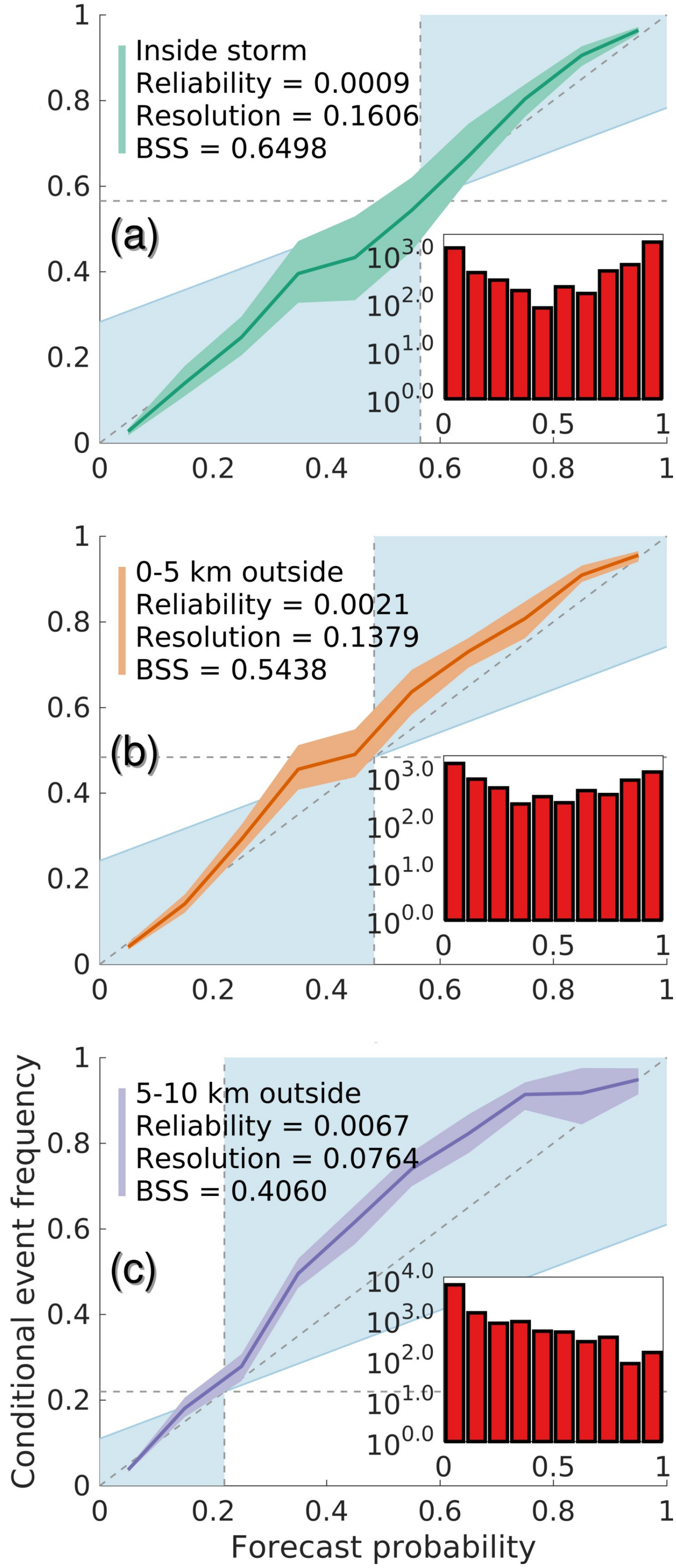

Fig. C5: As in Figure 10 but for 30–45-min lead time.

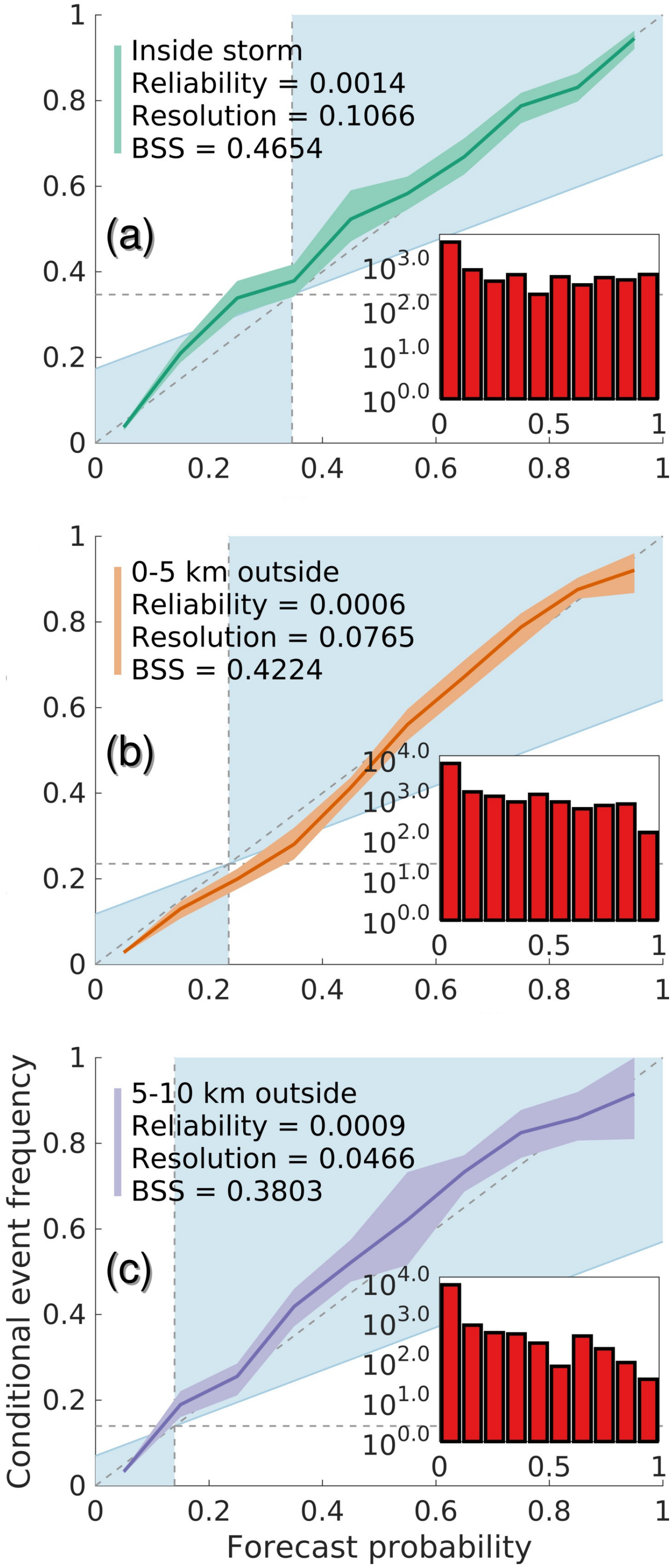

Fig. C6: As in Figure 10 but for 45–60-min lead time.