

Forecasting for Convectively Driven Wind Events at the Kennedy Space Center: Statistical Methods

Jared Rennie

Department of Atmospheric Sciences and Chemistry, Plymouth State University, Plymouth, New Hampshire

ABSTRACT

Forecasting for convective winds can be a rigorous task due to their small size. However many different methods, including those suggested by Loconto (2006) have been tested and put into operation. Unfortunately the probability of detection and false alarm ratios are not values we would like them to be. Using statistical methods described in class, a 13 year data set is used that contains 45 different variables as possible predictors. Different models are run via the R statistical program and they include logistic regression, Fisher's discrimination, classification trees, and bootstrapping. Once the model is created, a leave one out cross validation is performed along with calculating the misclassification rate, probability of detection, and false alarm ratio. Results show that all of the methods are better than the ones currently in use. Additionally, relevant variables such as the average relative humidity and K-Index are deemed significant in many of the models.

1) Introduction and Objectives

Convective winds are one of the major concerns at the Kennedy Space Center Complex, due to the low level wind shear it can produce. Nonetheless the 45th Weather Squadron is responsible for forecasting and issuing warnings for wind gusts generated by convective events that exceed 35 kts (Loconto 2006). This is done in order to avoid equipment costs, as well as ensure human safety.

One of the resources available to the forecaster is the launching of radiosondes into the atmosphere. Instrumentation attached to a weather balloon is launched on the complex at approximately 15Z (11am local time) everyday. Once data is collected, many atmospheric variables are calculated and subjectively looked at to determine if a convective event will occur.

Approximately 45 variables are calculated from one daily sounding. Determining a convective event from many different variables can be tedious to the forecaster, especially if time is limited. Currently, different methods of forecasting for convectively driven wind events have been suggested by Loconto (2006). However, the false alarm ratio (FAR) and probability of detection (POD) are both currently 50%. We would like to see a new method that will not only be limited to a few variables, but also have a large POD and a small FAR.

2) Data and Methodology

This paper will explore new forecasting techniques using statistical methods discussed in class. A dataset of 15Z sounding data during the months of May, June, July, August, and September over 13 years (1995 – 2007) are provided by Dr. Koermer. The

dataset is then inputted into the R statistical program and major quality control is performed. Each day was classified as either one of the following two events. If a peak wind gust was reported, then the day experienced a convective event and a value of 1 is given. If there was no peak wind gust (denoted as “NA”), then the day did not experience a convective event and a value of 0 is given.

Also, days that had either -9999 or NA needed to be removed, as R will not treat them correctly in the models. Additionally, one variable (maximum parcel level) was removed from the dataset, since more than half of the days had missing data. In total there were 1335 events, 797 where there was no convective gust, and 538 where there was a convective gust.

Once each statistical model was generated, the predict function in R was applied. This is done in attempt to see how the model would predict the event. Since it would be biased if the model was predicted using the same exact values, a leave one out (LOO) cross-validation is applied. Here, a fitting procedure is fitted n times with a sample size of $n-1$ (Wilks 2005). The predict() function is then applied n times where a value of either 0 or 1 is outputted. To make the workload easier, this process was automated.

Once there are observed and predicted values of 0 and 1, a confusion matrix is then constructed. An example is displayed in Table 1 below:

		<u>OBSERVED</u>	
		0	1
<u>PREDICTED</u>	0	a	b
	1	c	d

Table 1: Confusion matrix used to determine model performance

“a” is denoted as the true negative, where a convective event was not forecasted, and it did not occur. A false negative, where a forecast called for a convective event to not occur but it did occur, is noted by the letter b. When a forecast is made to prepare for a convective event, but it did not verify, then this is known as a false positive and it is noted by a “c.” Finally, when a forecast for a convective event verifies, this is then a true positive and is indicated by the letter “d.”

From this matrix, performance metrics can be calculated to see how well each model performs. The first is the misclassification rate, which is when a forecast for a specific event occurs, but the opposite happens. The equation is below:

$$Misclassification = \frac{b + c}{n}$$

where b is the false negative, c is the false positive and n is the total amount of events. Second, the probability of detection can be calculated. The POD describes the probability of a successful forecast. In other words, it will take the number of convective events and see how many were correctly forecasted by the model. The POD is given by the following:

$$POD = \frac{d}{d+b}$$

where d is the true positive and b is the false negative. Finally, the false alarm ratio (FAR) is calculated. It is similar to the misclassification rate, but only takes into account the times where a convective event was forecasted to occur, but did not occur. The FAR is given by the following:

$$FAR = \frac{c}{d+c}$$

where c is the false positive and d is the true positive.

3) Results

The following section will go into brief detail about each model, including description, variable selection, and model performance.

a) Logistic Regression

In general statistics, most models are run with linear regression. However one of the major assumptions in linear regression is that it assumes the data is normalized. This is not always the case, especially with some of the variables in our data set. Because of this, a logistic model can be run via the generalized linear model. It runs the same as ordinary linear models, as it fits the model with maximum likelihood estimation and provides hypothesis tests and an analysis of variance (ANOVA) table.

Variable selection was performed via the drop1 function. Creating a logistic model with all the covariates, the drop1 function was run with a Chi-Squared test to tell the user which variable was deemed the least statistically significant. The variable was then removed and the model was re-run. This was done recursively until all of the variables were statistically significant (p-values less than 0.05).

The final logistic model included the following variables as its predictors:

- Average Convective Available Potential Energy (AvgCAPE)
- Downdraft Wind (DWND)
- Precipitable Water (PW.in)
- Average Relative Humidity from Surface to 500hPa (RH to 500)
- 700-500 mb Lapse Rate (LR.C)
- Convective Condensation Level (CCL.mb)
- Convective Temperature (ConT.C)
- Lifted Index at 500mb (LI.C)
- Lifted Index at 700mb (LI700.C)
- Parcel Convective Inhibition (CnhJ.Kg)
- Minimum Equivalent Potential Temperature (MNTHe.K)
- Microburst Downdraft Potential Index - Version 2 (MMDPI2)

After cross validation, the model performance shows that the misclassification rate was 0.295, the POD was 0.646, and the FAR was 0.414.

b) Fisher's Discrimination and Classification

Discrimination can be performed on any model to try and separate observations from known populations. It does this by finding specific features that distinguish one classification from another and then attempts to draw a line between the two populations. That line can either be linear or quadratic. If the variance and covariance matrices are equal, then Fisher's linear discrimination is used. If the variance and covariance matrices are not equal, then the quadratic is used.

For our purposes the classifications are 0 (non-convective) and 1 (convective). Using both the `lda` and `qda` functions in R, it is determined that the means of each classification are nearly the same. However when the `predict` function is run, different performance numbers are shown. For `lda`, the misclassification rate was 0.29, POD of 0.63, and FAR of 0.4. For the `qda`, the misclassification rate, POD, and FAR were 0.36, 0.53, and 0.14, respectively.

c) Classification and Regression Trees (CART)

CART creates classification by dividing the entire sample into sub samples. These are based on the values of the predictors. The algorithm will search the dataset and maximize the significance or purity of a predictor and determine where and when a variable will be split into nodes. This is useful, because CART implements their own variable selection techniques. Also, because our response is categorical (either 0 or 1),

then these trees are known as classification trees. The user is able to select different parameters before running the algorithm, such as how many nodes to use and how large the tree should be. Once a tree is created, the user can also determine where to either adjust a node, or simply get rid of it. This is known as either snipping or pruning.

Three different CART trees will be shown, each with its own different techniques of creation, splitting, and pruning. The first uses the `tree()` function in R, which is considered as one of the oldest. The resulting tree includes 6 nodes and three variables, RH to 500, LI700 and the K-Index. The tree had a few unnecessary nodes, so after some manipulation, the final tree can be seen in Figure 1 below:

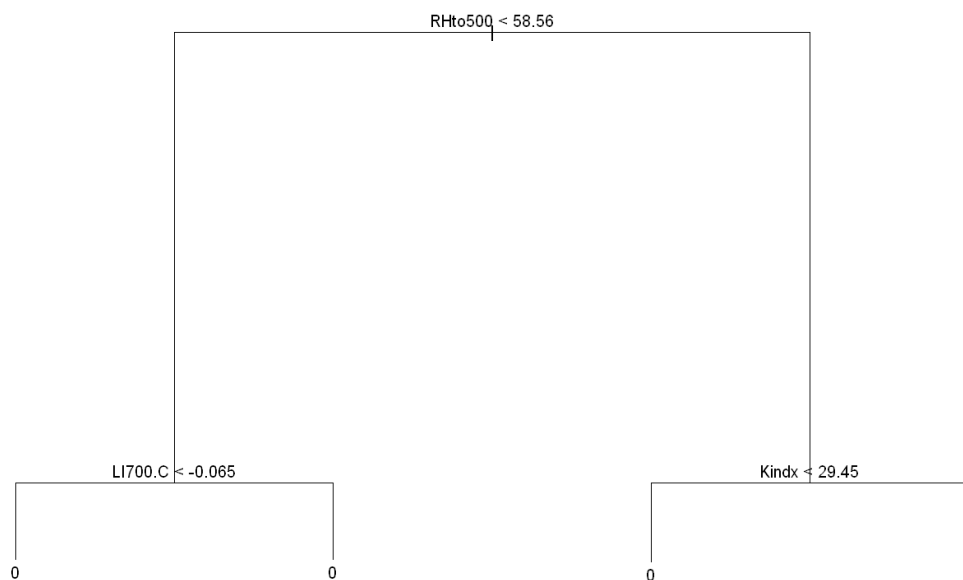


Figure 1: Classification tree of dataset using the `tree()` function in R

Technically the node with LI700.C could also be eliminated, since either result turns up as a non-convective event. Nonetheless the rest of the tree makes sense. If the average relative humidity to 500hPa is greater than 58.56% (meaning that the atmosphere is

saturated and is suspect to cloud formation), then it moves on to the next step. The K - Index is a stability parameter, where the higher the number, the better chance there is for a thunderstorm to occur. According to this tree, if the K-Index is greater than 29.45, then it will be a convective event.

This tree is useful, as it technically only uses two out of the forty-five original parameters from the dataset. Additionally, the model performance metrics enhances its usefulness, with a misclassification rate of 0.3, a probability of detection of 0.6, and a false alarm ratio of 0.3.

The second CART tree is derived from the `rpart()` function. This is a little more advanced than the `tree()` function, as it has different control parameters. This will cause the tree to be much larger. However the method of splitting up the tree is more useful. After manipulation, the following tree was created:

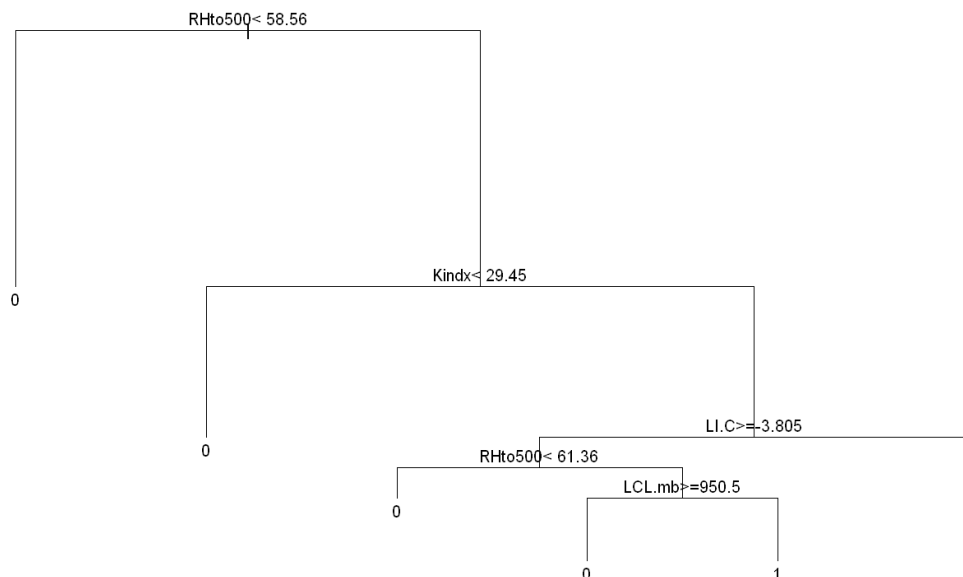


Figure 2: Classification tree of dataset using the `rpart()` function in R

The first two nodes of the tree are similar to the first tree in Figure 1. In this case however, there are more tests if the K-Index is greater than 29.45. It includes the lifted index (another parameter just like the K-Index in order to determine storm severity), the RH to 500 variable again, as well as the Lifted Condensation Level.

While this tree is a little more advanced than the first tree, the model performance metrics are still pretty solid. The misclassification rate and probability detection values are better (0.28 and 0.66, respectively), but the false alarm ratio is worse at 0.37.

The third and final tree was conducted using the party or ctree() function in R. Here, the algorithm uses conditional inference in attempt to prevent over fitting of the tree. The previous two trees do not test for statistical significance while this tree algorithm does. If the null can be rejected in favor of the alternative, then a node or split is created. Because of the use of statistical testing, no snipping or pruning is required.

The tree that is generated can be seen in Figure 3 below:

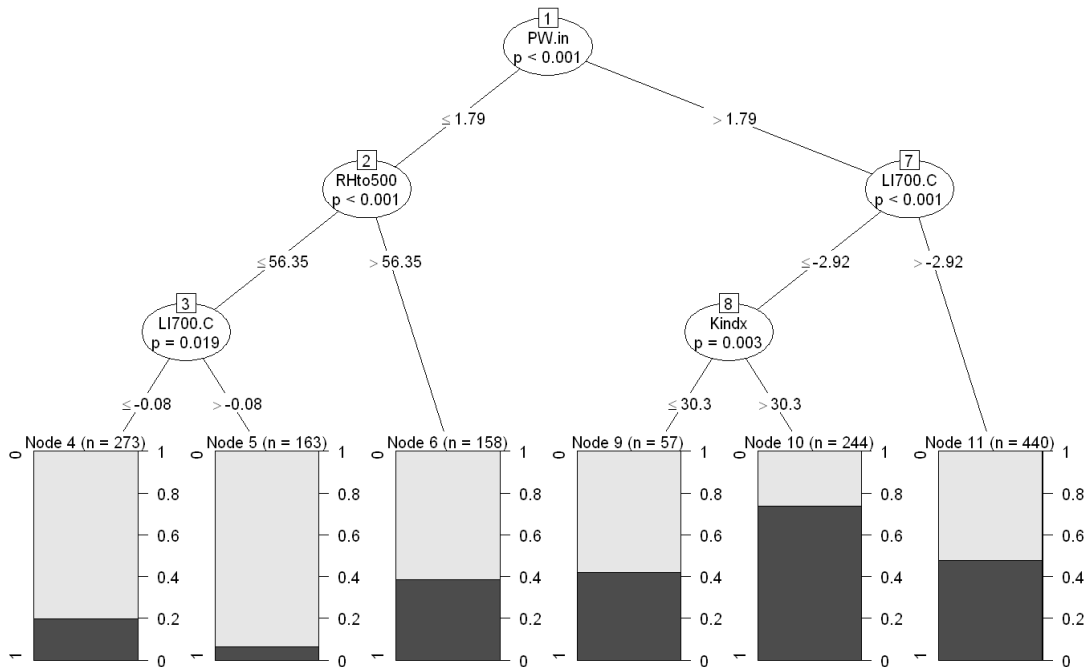


Figure 3: Classification tree of dataset using the ctree() function in R

Note that this tree uses almost the same variables as the previous two trees, except for the lifted condensation level. Additionally it starts with a new variable, the precipitable water. This indicates the amount of liquid water in a vertical column. Obviously enough the more precipitable water there is, the more rainfall there is. And given the proper surface buoyancy conditions (will not go into those details here), the wind gust can be stronger and therefore higher.

While this classification tree can be deemed as statistically significant, the model performance metric does not deem it as useful. While the misclassification rate is low (0.31) and probability of detection is large (0.73), the false alarm ratio is very high as well (0.66). While using this tree is good for predicting events, it will also have a high chance of incorrectly predicting events.

d) Bootstrapping

The process of bootstrapping is similar to CART, but it creates re-samples of data through the process of random sampling. This nonparametric procedure can eliminate some assumptions, but is also computationally intensive. There are three different methods of bootstrapping to talk about: bagging, boosting, and random forests.

In bagging, a bootstrap sample is taken and a CART model is applied. Once all the bootstraps are finished, the results are combined to produce a classification. The results are determined by a popular vote amongst the bootstraps. Like CART, certain parameters can be re-adjusted, including the number of bootstraps. Additionally, an out-of-bag (OOB) error is estimated. This is similar to the misclassification error, but it uses values that were not included in any of the bootstraps.

For this model, the OOB error is minimized when there are 400 bootstraps. Using this, a bagging model was created with the dataset, and a LOO cross validation was performed. The results show that the misclassification error was 0.31, the probability of detection was 0.62 and the false alarm ratio was 0.44.

Boosting is highly similar to bagging. The difference is the following: instead of random sampling, boosting will use an iterative algorithm to provide weighted versions of the learning sample. For example, classes that were identified in the previous step are given a higher weight on the next step, and classes that are not identified are then given a lower weight.

For our model, boosting was applied with fifty iterations, since the out of bag error tends to be minimized after that. After performing a LOO cross validation, results were nearly the same as bagging. The misclassification rate was 0.32, the probability of detection was 0.60, and the false alarm ratio was 0.44.

Random forest, like boosting, is similar to bagging except for one difference. The trees that are created from bagging are well correlated with each other. The process of random forest is to build trees that are not correlated with each other. Using random forests on our model, the misclassification rate was 0.32, the probability of detection was 0.61, and the false alarm ratio was 0.46.

Additionally, variable importance plots can also be shown to measure the improvement at each split for each of the predictors. In simple terms, it will produce the importance of each predictor in the random forest model. Figure 4 below shows the variable importance plot for our random forest model. Note that the predictors with the highest of importance have also been used in the previous CART models

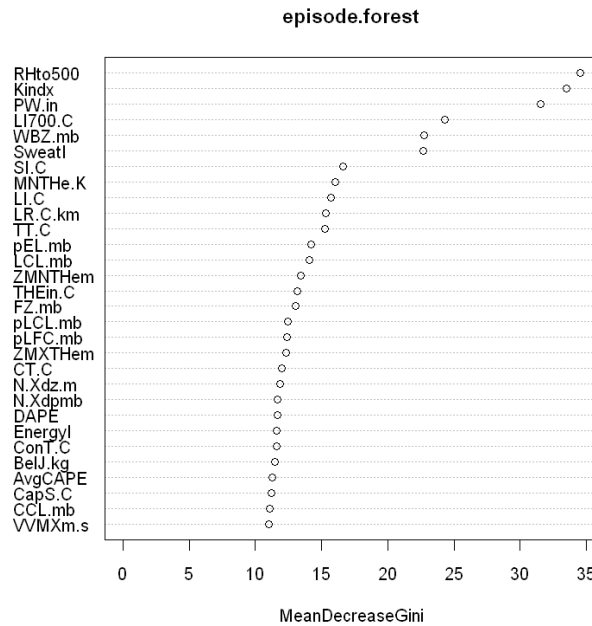


Figure 4: Variable Importance Plot for random forest model in R

4) Discussion / Conclusions

The performance metrics for all nine models are listed in Table 2 below:

	Misclass	POD	FAR
episode.logistic	0.29	0.64	0.41
episode.lda	0.29	0.63	0.40
episode.qda	0.36	0.53	0.14
episode.tree	0.30	0.60	0.30
episode.tree2	0.28	0.66	0.37
episode.tree3	0.31	0.73	0.66
episode.bagging	0.31	0.62	0.44
episode.boosting	0.32	0.60	0.44
episode.forest	0.32	0.61	0.46

Table 2: Misclassification error, probability of detection, and false alarm ratio for all models used in this research

Looking at the misclassification errors, values are low and tend to have a small variance, between 28% and 36%. The lowest value is with the second CART tree using rpart() and the highest value is using Fisher’s quadratic discrimination. However, overall the values are fairly low, which is what we would like to see.

The probability of detection values are all above 50%, which are improvements from the methods suggested by Loconto (2006). The best value is with the CART tree using the `ctree()` function (`episode.tree3`) at 73%. However this is also associated with the highest false alarm ratio, so this method is no good. The next highest value is then the CART tree using `rpart()` function (`episode.tree2`) at 66%. The lowest probability of detection is Fishers quadratic discrimination, at 53%.

The lowest false alarm ratio is associated with Fisher's quadratic discrimination at 14%. That sounds very good, however this model is also associated with the highest misclassification error, and the lowest probability of detection. Therefore this model should not be used either. The next model with the lowest false alarm ratio is then the first CART using the `tree()` method (`episode.tree`), at 30%. The highest false alarm ratio, as noted earlier was with the third classification tree using `ctree()` at 66%.

When choosing the best model out of the nine, I think about the general meteorologist. Most meteorologists like to see not only simplicity, but also visual aids. Sometimes the need to make a forecast is quick, and a lot of thought process needs to be taken out of the operation. Therefore I choose the CART models as the best methods. On the other hand, it cannot be too simple, because it may not include everything needed to make a proper forecast.

Therefore, I choose the second classification tree (`episode.tree2`) as the best model to come out of this project. This does have the lowest misclassification error and the highest probability of detection, while at the same time, conserving a fairly low false alarm ratio. Additionally, as seen in Figure 2 above, it uses four different variables (RH

to 500, K-Index, Lifted Index, and Lifted Condensation Level) instead of just two with the first classification tree.

Another thing to note from these results is that both the RH to 500 and K-Index variables were used fairly often. This makes a lot of sense. First, Florida is in a very warm, moist, tropical airmass. If this moist layer is deep enough (say from the surface to 500mb), a lot of clouds can be created (if there is enough lift) and hence a lot of water can be precipitated. Additionally, because of the small scale features of thunderstorms in Florida, they tend to be known as airmass type thunderstorms. The K-Index was created solely for the purpose of airmass type thunderstorms. So seeing these two variables makes sense to the user.

5) Summary

This project proposed to create different methods for forecasting convective winds at the Cape Canaveral Complex in the state of Florida. After receiving a 13 year dataset and performing rigorous quality control, many statistical methods were performed, as well as calculating variables to determine the performance of the model. Most of the results deemed better than 50/50 guessing, with low misclassification rates. It was determined that the second classification model deemed as best fit for forecasting convective wind events.

6) References and Acknowledgements

Loconto, A.N., 2006. Improvements of Warm-Season Convective Wind Forecasts at the Kennedy Space Center and Cape Canaveral Air Force Station. M.S. Thesis, Dept. of Chemical, Earth, Atmospheric and Physical Sciences, Plymouth State University, Plymouth, NH.

Wilks, D. S., 2005: *Statistical Methods in the Atmospheric Sciences, Volume 91, Second Edition*. Academic Press, 648.