Predicting Incidence of West Nile Virus in Chicago

Christopher Aden

May 19, 2015

1 Introduction

West Nile virus (WNv) is a virus transmitted through mosquitos. If a mosquito is infected and takes a human blood-meal, there is about a 20% chance that the individual will become infected, with symptoms including fever, neurological damage, or death. As mosquitos lay their eggs in still water and require warmer climate, WNv tends to occur most often in the summer time [9].

The city of Chicago experienced its first cases in 2002, and within two years, The Department of Public Health (CDPH) had established surveillance programs to track the spread of WNv, and control programs to reduce the incidence. As part of their efforts to track the virus, CDPH has released data and asked the community to predict the conditions that lead to WNv. The data comes to us in the form of mosquito traps, containing various species of mosquito, each tested for the presence of West Nile virus. The traps are mapped to a nearby address and labeled with latitude and longitude.

Later, CDPH determines which traps had high numbers of mosquitoes and sprays an insecticide in an effort to lower the number of infected mosquitoes. This data is given as a geo-tagged set of time-stamps (latitude, longitude, date).

Finally, Chicago has two weather stations at its airports. These stations provide valuable daily data about weather conditions, including temperature, precipitation, wind speed and direction, and so on.

Given weather, trap, and spray data, this paper will attempt to answer four central questions:

1. Controlling for location effects, is spraying an effective method of reducing the incidence of West Nile virus?

2. What are the main weather conditions that influence West Nile virus incidence and how do they influence it?

3. Given all data, can we accurately predict the probability that a particular mosquito species in a given trap will have West Nile virus?

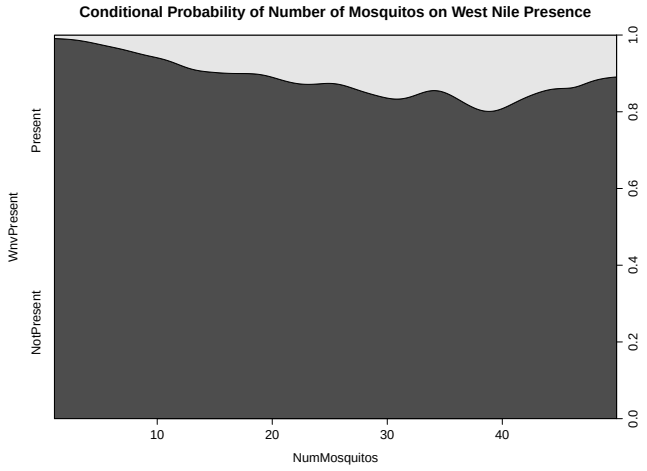
4. Can we develop a statistical tool that has a useful sensitivity to detect West Nile virus, while still keeping the specificity high?

The first two questions are inferential in nature, while the third is a prediction problem. The paper is divided accordingly, as the methods used are largely independent.

2 Inference and Exploratory Analysis

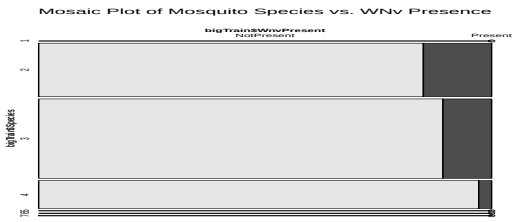
While predictive modeling has come into popularity in the last several years, there is still a need for inference to determine the direction and presence of significant effects. In this section, we explore how each predictor (trap, spray, and weather) affects the incidence of WNv.

We begin with an exploratory analysis of the trap data, which requires very little "data wrangling" to analyze. The trap data has $10,506$ rows (each row *roughly* corresponds to one species in one trap on one day, though a row is split into multiple if there are more than 50 mosquitoes of one species in a trap on a given day), 136 traps, accounting for $135,039$ mosquitoes, of which $14,519$ have WNv (prevalence $= 0.108\%$). There is sufficient a priori subject knowledge that all of the trap covariates are meaningful in explaining WNv, though potentially not all in conjunction. For example, mosquitoes lay their eggs in stagnant water. Traps located near stagnant water sources would thus be more likely to have more mosquitoes. The next logical jump would be to determine if the probability of WNv is related to the number of mosquitoes. A conditional plot looks at the conditional probability of WNv as a function of the number of mosquitoes. Indeed, it would appear that the probability of West Nile does go up as the number of mosquitoes increases, but decreases as the number of mosquitoes approaches the row maximum of fifty.



Do certain species of mosquito carry West Nile more frequently than others? Since our outcome (WNv present or not) and predictor (species) are categorical, mosaic plots are employed. If all species have the same incidence, the dark gray boxes, indicating WNv

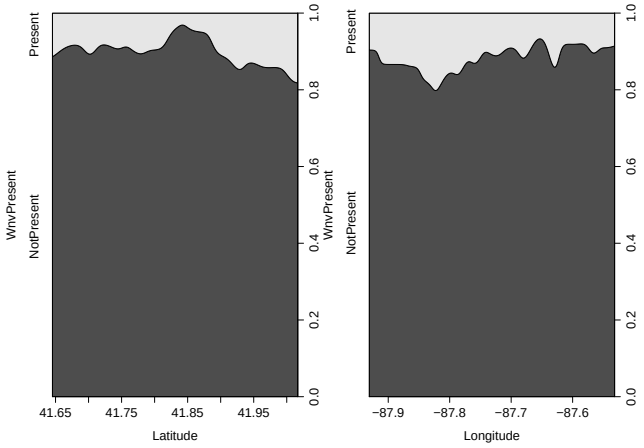
frequency, ought to be the same width (the height shows the species' relative frequencies). A table with the species integer mapping is provided with the mosaic plot, and it's quite clear there are differences in the prevalence West Nile virus between species.



Key	Species
1	*C. Erraticus*
2	*C. Pipiens*
3	*C. Pipiens* or *C. Restuans*
4	*C. Restuans*
5	*C. Salinarius*
6	*C. Tarsal*
7	*C. Territans*

In fact, the only three groups with WNv present were 2, 3, and 4, which maps to either *C. Pipiens* or *C. Restuans*! Previous studies confirm this result, citing *C. Pipiens* [12], *C. Restuans* [10], and *C. Salinarius* [10] as confirmed vectors of WNv. As a sanity check, a Monte Carlo-based χ^2 test of association between WNv presence and Species is staggeringly significant ($\chi^2 = 2472$, $p \ll .00001$), with the p-value limited by the number of MC simulations, indicating a likely relationship between species and WNv incidence.

Similarly, location may play a role. As we expect, conditional plots reveal that there is a clear effect of latitude and longitude, but not a clear linear pattern on either.



Latitude and Longitude Conditional Density Plots

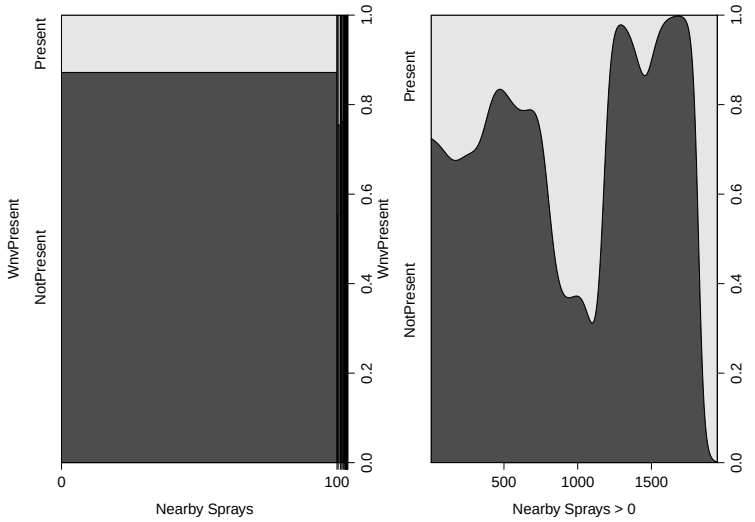
We move on to spray data. This was quite a controversial piece in the Kaggle competition. Spraying data is provided only for years that are not in the set of data being judged in the competition (2011, 2013). The Inference chapter focuses only on this subset of the data that

includes the spray data, while the prediction chapter will focus on both. As the administrator of the competition wrote in a reply about the insecticide spray data for the years in the testing set, "as far as we know, they haven't been super effective in controlling the spread of WNV" [4] and having the sprays be tied to prior West Nile surveillance, the inclusion of spray data "could lead to data leakage, therefore, spray data in the test years are not provided" [5]. The data proved *exceptionally* difficult to manage. Each spray is tagged with a latitude, longitude, and time stamp. It is plausible that sprays closer in spatial and temporal proximity to a given trap (on a particular day) are more effective than sprays that happened in the previous year, or on the opposite side of town. The sophisticated way for dealing with this would be to apply a temporal-spatial model that allows for a correlation between space and time that decays exponentially (or polynomially). This is computationally and theoretically beyond the scope of this analysis. As a cheap proxy, we assume that the effective radius of a given spray is no more than two miles (the speculated range of a given mosquito), and that the spray effect has completely diminished after two weeks. This approach then counts how many sprays have occurred in the last two weeks in a two mile radius, which is *still* computationally difficult, given that there are $14,294$ sprays! We pare down the size of the problem by subsetting our trap data to only the years we have spray data for, which gives $4,446$ latitude-longitude pairs. Euclidean distance is an extremely poor metric for map distances, because the Earth is not a flat surface. Relative errors for using Euclidean geometry can be quite high. To overcome this, we use the Haversine formula, designed especially for computing distances on spheres. For two latitude-longitude pairs (ϕ_1, λ_1), (ϕ_2, λ_2), the Haversine Formula gives d as

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right),$$

where r is the radius of the Earth ($3,959$ mi). We then compute all $(14,294)(4,446) = 63,551,124$ distances and determine which are less than two miles, then filter only the recently-occurring sprays. The result is reduced one more time to give an integer of the number of spatiotemporally-proximal sprays for each trap-day.

The vast majority of traps have not experienced a recent spray (94.6%). Histograms of nearby sprays with and without the zero sprays are included in subsection two of the Appendix. This is important for analysis when we look at the conditional density plots. We won't see a change in the conditional density until the 95^{th} percentile of sprays, distorting the picture. Excluding the zeroes, there is a clear difference in the density over sprays, but not in any clear way.



Zero included (left) and excluded (right) Proximal Spray Count Conditional Densities

Weather data is much simpler. Each of the two weather stations (located at O'Hare International Airport and Midway International Airport) recorded weather data every day over the course of the study. We continue to focus only on the data we have sprays for (years 2011 and 2013, $n = 55,243$). We begin by averaging the two weather stations' observations together (the differences between the two stations is relatively small), then merging the weather data to the trap data through the date on each weather record. Collected weather data and descriptions are available through NOAA's Quality Controlled Local Climatological Data documentation (`http://www.ncdc.noaa.gov/qclcd/qclcddocumentation.pdf`). We will skip description of the exploratory analysis on the weather data. Prior work[9] suggests that air temperature and low humidity play very significant roles in human infection rates of West Nile virus, and potentially other factors could play significant roles, too. Logistic regression models that incorporate all variables mentioned in the NOAA QCLCD documentation, our spray data, and our trap data are included in the appendix. One model includes proximal sprays as a continuous term, while the other dichotomizes it to whether it is nonzero or not. However, both models lack parsimony. Additionally, it is very likely that the predictors are highly correlated, due to our inclusion of an interaction term for longitude and latitude of the trap. The interaction allows the odds of WNv to increase not just North-South or East-West, but in more complicated directions.

At this point, we pursue model selection, which requires caution. To avoid over-fitting (reporting a model that is more significant than in reality by training the model on the same data as the test set), we train on 80% of the observations and withhold 20% to test the model's validity. We perform ridge regression on the dichotomized model to reduce the collinearity, but this proves somewhat futile. Ridge regression performs very admirably at reducing the variance inflation factors (see table in Appendix), but the model still has unacceptably large collinearity. What we really need is to drop terms entirely from the model. This is where

we employ ElasticNet [13], a hybrid penalization method that minimizes

$$\frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda\left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right],$$

where α determines the balance between ridge regression and lasso, and λ is the tuning parameter. ElasticNet is a combination of L2 (ridge regression) and L1 (lasso) penalties, allowing terms to drop from the model completely, while avoiding lasso's drawback of keeping only one similar predictor in the final model. `R`'s implementation, `glmnet`, optimizes λ for a fixed α. Commonly, α is kept close to either 0 (for ridge) or 1 (for lasso). We leave it as .9 to mimic lasso to get a simplistic model. Using cross-validation, we find the λ that minimizes the deviance, then run lasso with a λ that is one standard deviation larger for increased parsimony. The terms of the model reduced to zero are dropped, and the reduced model is tested on the withheld data ($n = 11,049$).

This model is far easier to read and terms are significant even in the testing data. Some terms are not statistically significant, but the testing data set is also *much* smaller than the training set. The variance inflation factors have largely stabilized, except for StationPressure and SeaLevel, which measure the air pressure at the weather station and sea level, respectively, so collinearity is to be expected. Removing the sea level pressure makes the station pressure extremely significant, and the VIFs stabilize. In the face of the ElasticNet's recommendation, it may be prudent to keep only the weather station pressure.

	Estimate	SE	z-value	p-value
(Intercept)	-470.5712	44.66	-10.54	$\ll 0.0001$
isSpecies1247TRUE	-0.1289	0.06	-2.05	0.0405
Longitude	-3.7336	0.60	-6.21	$\ll 0.0001$
Year2013	0.6037	0.11	5.70	$\ll 0.0001$
Month7	2.3112	0.42	5.51	$\ll 0.0001$
Month8	4.6864	0.42	11.28	$\ll 0.0001$
Month9	4.4322	0.42	10.66	$\ll 0.0001$
Tmax	0.0712	0.01	8.34	$\ll 0.0001$
HeatDegreeDay	0.0781	0.02	3.19	0.0014
StationPressure	3.9241	0.70	5.60	$\ll 0.0001$
ResultSpeed	0.0980	0.02	5.74	$\ll 0.0001$
anyNearbySpraysTRUE	0.5445	0.11	5.06	$\ll 0.0001$
Latitude:Longitude	-0.0042	0.005	-0.87	0.3859

Term	VIF
isSpecies1247	1.04
Longitude	4.08
Year	1.89
Month	2.07
Tmax	3.97
HeatDegreeDay	2.95
StationPressure	4.79
ResultSpeed	2.98
anyNearbySprays	1.08
Latitude:Longitude	4.04

We can also try a generalized additive model with a tensor product; a cheap trick to allow a GAM to fit an interaction. Looking at conditional density plots gives us an idea which terms are highly nonlinear. It turns out nearly all of them, though HeatDay is poorly-behaved. Output from the GAM approach is attached in the appendix. This approach is far superior at modeling the latitude and longitude, as the linearity assumption in the logistic regression is too strict for something like a location, but it lacks the interpretability of logistic regression. The additional terms also mean we lose power; nearly all coefficients are now not significant, except for the spray effect and location effect. A less nuclear approach would be to only make smooth terms for the location and leave everything else as linear, despite their nonlinearity. While this would not capture the nuances of their relationships, we would not take such a large penalty to our degrees of freedom.

Finally, it is quite likely that within a specific trap, there will be correlations. This is due not only due to similar location, but that mosquitoes in the same trap may have behaved similarly in contracting WNv themselves. We attempt to model this with a random effect for the specific trap. Since our goal is to make a statement about the average trap, the marginal random effects model is most appropriate. We choose an exchangeable correlation structure (all mosquitoes within a trap share one common correlation) for computational feasibility, though at this sample size, sandwich estimators asymptotics are valid, and our choice of correlation structure is less important. Output is attached in the appendix, and shows a strong effects from longitude, year, month, and proximity to an insecticide spray. Species and weather conditions were not significant in this model.

3 Prediction

Statistical inference has its place in the planning room, determining which predictors are most meaningful in explaining the odds of WNv, but prediction also serves a vital function. While we produced a model earlier in the inference section and analyzed the predictors that most impacted incidence of West Nile virus, the aforementioned generalized linear models are rarely the best tools if the only goal is prediction, as they are forced to obey the rigid linearity constraint (through the linear predictor). In this section, we disregard any notion of statistical significance and instead strive to achieve a high rate of predictive accuracy. The realistic motivation for this section is that once we have collected weather and trap data, we want the most accurate possible prediction for whether a given mosquito has West Nile virus.

There are two main aims for this section. First, we will create models that try to predict WNv as accurately as possible. Second, we will evaluate metrics subjectively, assessing whether they meet our needs for sensitivity and specificity jointly.

As a comparison of the more modern machine learning methods used in the section, each subset and each metric will be compared against a logistic regression and ElasticNet model. The simplicity of both approaches lends itself well to interpretability, and it may be useful to use them for prediction. The primary computational tool employed is the `R` package `caret`, which has a parallel grid-search optimizer for algorithm parameters and has cross-validation and output methods built-in. In each case, metrics are estimated using 5-fold cross-validation, training on 80% of the data, then estimating the metric with the remaining 20%. A short description of each predictive model used is as follows:

1. Logistic Regression: A full logistic regression, with no model selection.

2. ElasticNet Logistic Regression: A full logistic regression followed by a grid-search for optimal (α, λ) to use in an ElasticNet.

3. RandomForest[1]: A tree-based ensemble method where a combination of trees, each one taking a random subset of the data *and* a random subset of the *predictors*, are averaged to create a consensus classifier.

4. Adaptive Boosting (AdaBoost) [11]: An ensemble method that combines the information from trees that show very little improvement over guessing (like a decision tree with only one split), then tunes subsequent trees to perform better on the samples that the previous trees misclassified.

5. Gradient Boosting Machines[3]: A more general class of ensemble methods that includes AdaBoost. GBMs allow for more general (differentiable) loss functions, and optimizes using gradient descent with a modification to control the rate of descent.

Since the aim of the section is to optimize our classification rate, we will look at three different subsets of the data:

1. Use the trap and weather data from all available years (2007, 2009, 2011, 2013).

2. Use the trap, weather, and spray data from years in which we have insecticide spray data (2011, 2013) and see how well it validates against held-out data from the same time period (to judge how effective spray data is for classification).

3. Use trap and weather data from just the years we have spray data. The goal is to answer how much worse model quality would be without spray data.

This model-free approach would usually be quite problematic, due to concerns of overfitting and data-snooping, but we solve this problem by snooping on the training set and validating on the test set. The ensemble methods also do their own built-in withholding of data, either by operating on weak learners, or using only a subset of the data at a time. These more advanced machine learning techniques are never given a full glimpse of the data at any one time. As a naive criterion, we might seek to maximize the AUC or minimize the misclassification rate (sum of the false negative and false positive rates). A 5-fold cross-validated confusion matrix shows us the proportional breakdown of negatives and positives for each method.

Method	Data	True Neg	False Neg	False Pos	True Pos
GLM	AllYears, Weather	89.10	10.50	0.10	0.30
ElasticNet	AllYears, Weather	89.10	10.60	0.10	0.10
RandomForest	AllYears, Weather	88.30	4.00	0.90	6.70
GradientBoosting	AllYears, Weather	87.80	6.20	1.50	4.60
AdaBoost	AllYears, Weather	88.40	9.20	0.80	1.60
GLM	SprayYears, Weather	86.10	12.50	0.50	0.90
ElasticNet	SprayYears, Weather	86.20	12.70	0.40	0.70
RandomForest	SprayYears, Weather	85.00	3.20	1.60	10.20
GradientBoosting	SprayYears, Weather	84.40	5.00	2.20	8.40
AdaBoost	SprayYears, Weather	84.60	8.80	2.00	4.60
GLM	SprayYears, SprayWeather	86.10	12.40	0.50	1.00
ElasticNet	SprayYears, SprayWeather	86.10	12.50	0.50	0.90
RandomForest	SprayYears, SprayWeather	85.10	3.30	1.50	10.10
GradientBoosting	SprayYears, SprayWeather	84.50	5.00	2.10	8.40
AdaBoost	SprayYears, SprayWeather	84.50	9.00	2.10	4.40

The models all achieve remarkable accuracy, but they do so by predicting no WNv almost every time! This is a well-studied problem with prediction problems. The problem is that the metric is a poor choice to optimize when the classes (presence and non-presence of WNv) are not close to 50-50. Realistically, we are willing to take a hit in overall accuracy (losing specificity) if it means gaining much more sensitivity. This is because a model that cannot predict a positive when a mosquito actually has WNv has no predictive power. The problem is solved by instead choosing one of two different metrics that attempts to punish classifiers that try to "cheat" and use the unbalanced classes to their advantage.

1. F_1 Score[8]: $F_1 = 2 \cdot (\text{PPV} \cdot \text{Sens}) \,/\, (\text{PPV} + \text{Sens})$, is a metric that strives to have a high sensitivity (probability of predicting WNv when a mosquito actually has WNv), but weights it according to the positive predictive value (probability of a mosquito having WNv when I have predicted it to be present). With the GLM model using only weather data on all years, for example, F_1 would be $F_1 = 2 \cdot ((.03/.04) \cdot (0.3/10.8))/((.03/.04) + (0.3/10.8)) = 0.054$. Predicting all mosquitoes as WNv-free would give an F_1 score of 0, so the classifier is not fairing well.

2. Balanced Accuracy[2]: The easier to understand of the two metrics, balanced accuracy is the mean of the specificity and sensitivity. In the accuracy-optimizing case, the models perform well by achieving a high specificity and sacrificing their sensitivity. With balanced accuracy, this cannot be done without taking a large penalty. With the GLM model using only weather data on all years, for example, balanced accuracy would be $BA = ((89.1/89.2) + (0.3/10.8))/2 = 0.513$. Predicting everyone to belong to one class results in a balanced accuracy of .50, so this metric does indeed correct the greediness of optimizing for total accuracy.

Re-running these models, optimizing now for F_1 and BA, we come up with much more realistic confusion matrices.

Method	Data	True Neg	False Neg	False Pos	True Pos	F1
GLM	AllYears, Weather	89.10	10.50	0.10	0.30	0.05
ElasticNet	AllYears, Weather	89.10	10.50	0.10	0.20	0.04
RandomForest	AllYears, Weather	88.30	4.00	1.00	6.80	0.73
GradientBoosting	AllYears, Weather	87.90	6.30	1.40	4.50	0.54
AdaBoost	AllYears, Weather	88.40	9.10	0.80	1.70	0.26
GLM	SprayYears, Weather	86.10	12.50	0.50	0.90	0.12
ElasticNet	SprayYears, Weather	86.50	12.90	0.00	0.50	0.07
RandomForest	SprayYears, Weather	85.20	3.40	1.40	10.00	0.81
GradientBoosting	SprayYears, Weather	85.30	10.50	1.30	2.90	0.33
AdaBoost	SprayYears, Weather	84.60	8.80	2.00	4.70	0.47
GLM	SprayYears, SprayWeather	86.10	12.40	0.50	1.00	0.13
ElasticNet	SprayYears, SprayWeather	86.40	12.80	0.20	0.60	0.09
RandomForest	SprayYears, SprayWeather	85.10	3.30	1.50	10.10	0.81
GradientBoosting	SprayYears, SprayWeather	85.40	10.20	1.20	3.20	0.36
AdaBoost	SprayYears, SprayWeather	84.50	8.70	2.10	4.70	0.47

Confusion Matrices for F_1-based optimization

Method	Data	True Neg	False Neg	False Pos	True Pos	BA
GLM	AllYears, Weather	89.10	10.50	0.10	0.30	0.51
ElasticNet	AllYears, Weather	89.10	10.50	0.10	0.20	0.51
RandomForest	AllYears, Weather	88.20	3.90	1.00	6.80	0.81
GradientBoosting	AllYears, Weather	87.80	6.60	1.50	4.10	0.68
AdaBoost	AllYears, Weather	88.40	9.10	0.80	1.60	0.57
GLM	SprayYears, Weather	86.10	12.50	0.50	0.90	0.53
ElasticNet	SprayYears, Weather	86.00	12.50	0.50	0.90	0.53
RandomForest	SprayYears, Weather	85.00	3.20	1.50	10.20	0.87
GradientBoosting	SprayYears, Weather	84.50	5.20	2.10	8.30	0.80
AdaBoost	SprayYears, Weather	84.60	8.80	2.00	4.60	0.66
GLM	SprayYears, SprayWeather	86.10	12.40	0.50	1.00	0.53
ElasticNet	SprayYears, SprayWeather	86.10	12.50	0.50	0.90	0.53
RandomForest	SprayYears, SprayWeather	84.90	3.20	1.70	10.20	0.87
GradientBoosting	SprayYears, SprayWeather	84.50	5.00	2.10	8.40	0.80
AdaBoost	SprayYears, SprayWeather	84.50	9.00	2.10	4.40	0.65

Confusion Matrices for BA-based optimization

No one method particularly out-performs the rest, but GLMs and ElasticNets lag behind. The more sophisticated models took a hit in overall accuracy with these two metrics, but made up for it in improved positive-sample classification. In the end, that was what we cared about all along. Fitted to their respective three problems without cross-validation (to be used in the Kaggle competition), the ROC curves for the three data scenarios were created, and are attached in the Appendix.

4 Conclusion

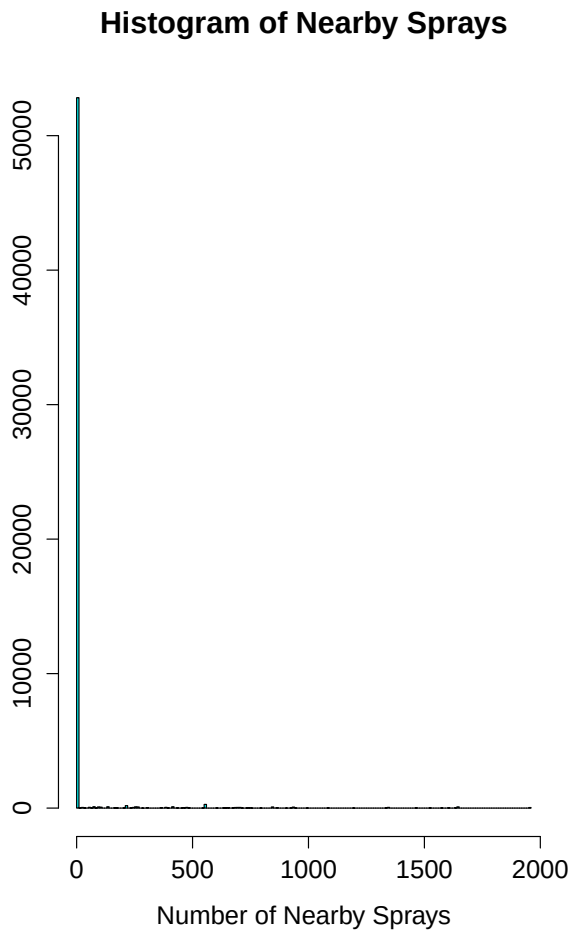
On the subject of inference, regularization allowed us to separate the wheat from the chaff, finding a very significant model, that was parsimonious and intuitive to explain. Previous weather findings from other scientists were reinforced as our model demonstrated that hot and arid conditions are most amenable to West Nile virus. There was a clear location effect, and future work would need to cover a spatial analysis, but smooth with GAM was still able to find a meaningful relationship between location and the presence of WNv.

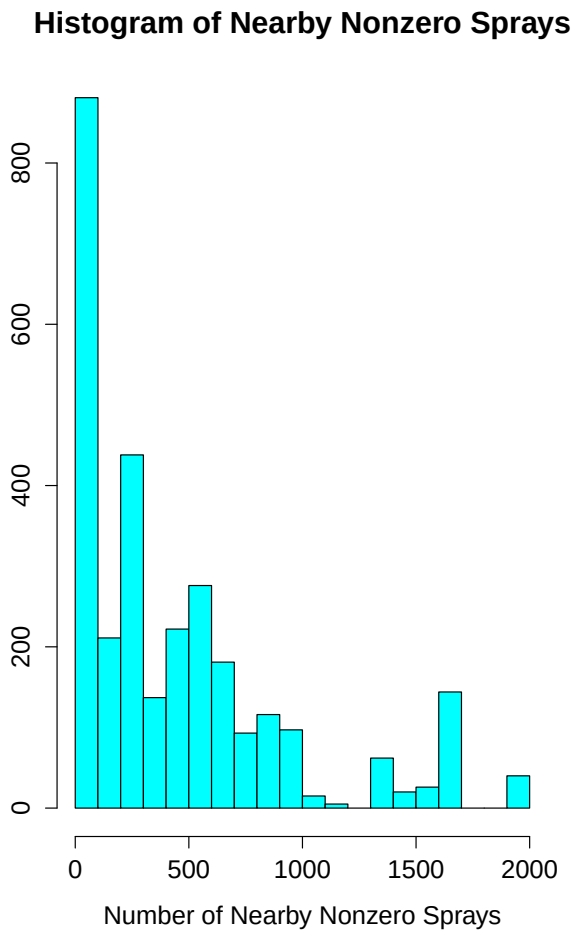
On the topic of the competition administrator's initial comments about the number of proximal sprays not being effective at controlling the spread, the evidence is inconclusive without more sophisticated tools. The logistic model demonstrates that the sprays are at least being aimed in the right places, as there is an increased odds of WNv in places that are being sprayed, but this cannot speak to the effectiveness of the insecticide.

On the subject of prediction, high AUC's were explained by uneven frequencies. More realistic tools like F_1 and Balanced Accuracy were still able to salvage insight from the data, and produced very high scores, nonetheless. These models have the advantage of high sensitivity, which the accuracy-optimized models did not. Future work would focus on the spatiotemporal aspect of the data and attempt to draw conclusions about the spread of WNv over time using more advanced methods than GLMs and GLMMs.

5 Appendix

5.1 Figures and Tables





Histograms of Number of Nearby Sprays and Number of Non-Zero Sprays

	Unpenalized	Unpenalized Boolean	Ridged Dichotomous ($\lambda = 3.596$)
Latitude	1092549.8	1089602.8	16.2
Longitude	218429.4	217844.1	5.6
Tavg	35.2	35.3	33.7
DewPoint	224.5	224.5	205.0
WetBulb	324.4	325.0	296.0
Heat	3.6	3.6	3.5
PrecipTotal	4.9	4.9	4.8
StnPressure	179.0	179.4	168.0
SeaLevel	184.1	184.6	172.5
ResultSpeed	19.2	19.2	18.8
ResultDir	1.8	1.8	1.8
AvgSpeed	17.7	17.9	17.6
NearbySprays	1.0	5.2	5.0
as.factor(Month)7	5.2	3.3	3.3
as.factor(Month)8	3.3	3.1	3.1
as.factor(Month)9	3.1	2.4	2.4
as.factor(Year)2013	2.3	1360.9	583.0
as.factor(Species)2	1360.9	2742.7	1174.1
as.factor(Species)3	2742.7	2352.8	1007.4
as.factor(Species)4	2352.8	13.8	6.5
as.factor(Species)5	13.8	2.3	1.5
as.factor(Species)6	2.3	29.0	13.0
as.factor(Species)7	29.0	1.1	1.1
Latitude:Longitude	2046847.0	2041333.0	27.9

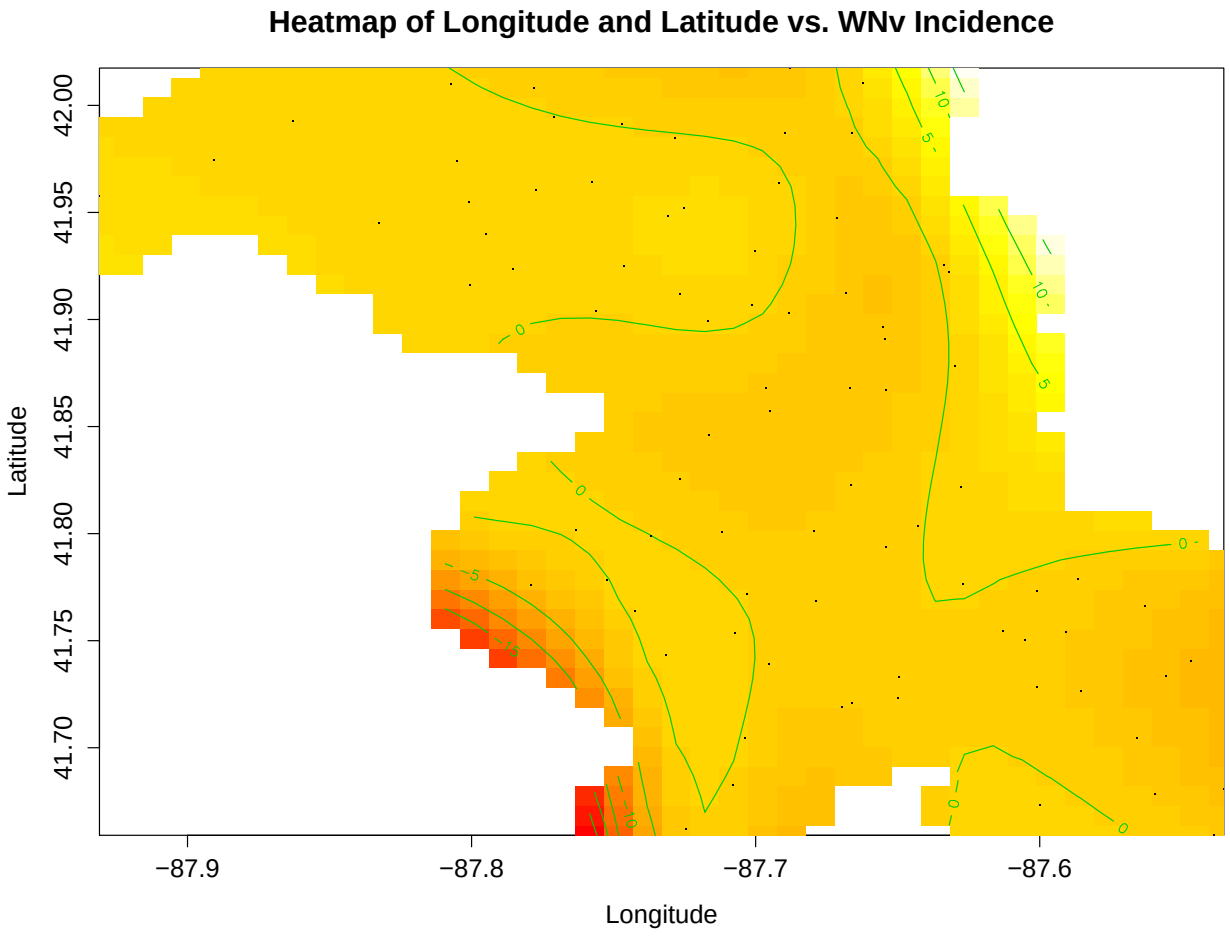
Variance Inflation Factors for Logistic Regression with Continuous Spray Count, Dichotomized Spray Count, and Dichotomized Spray Count with Ridge Regression

	Estimate	Std. Error	z-value	p-value
(Intercept)	-41.434	450.226	-0.092	0.927
isSpecies1247TRUE	-0.103	0.067	-1.550	0.121
Year2013	-1.052	9.158	-0.115	0.909
Month7	28.549	463.045	0.062	0.951
Month8	38.442	464.433	0.083	0.934
Month9	31.644	464.695	0.068	0.946
HeatDegreeDay	12.919	15.203	0.850	0.395
anyNearbySpraysTRUE	0.343	0.118	2.893	0.004

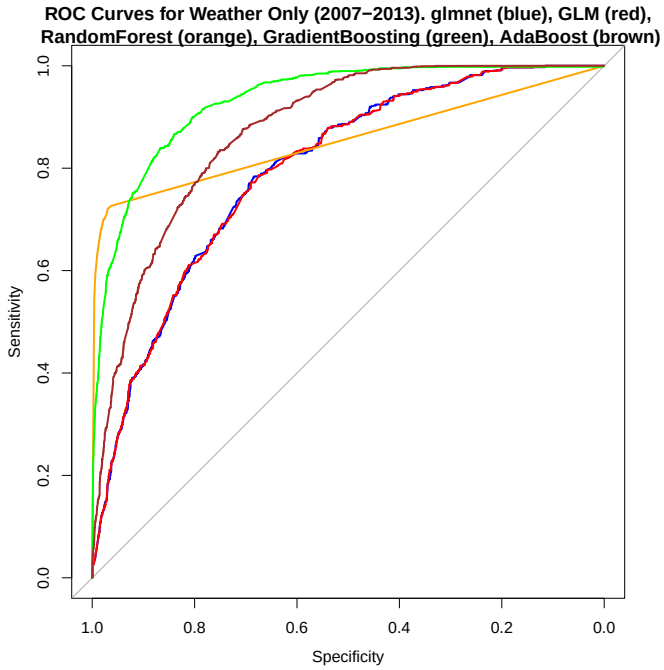
Output of Linear Terms from GAM on Holdout Data

	edf	Ref.df	χ^2	p-value
te(Latitude,Longitude)	23.804	23.981	345.382	$\ll 0.0001$
s(Tmax)	6.058	6.150	3.061	0.815
s(StnPressure)	5.403	5.604	2.375	0.852
s(ResultSpeed)	8.179	8.190	13.190	0.114

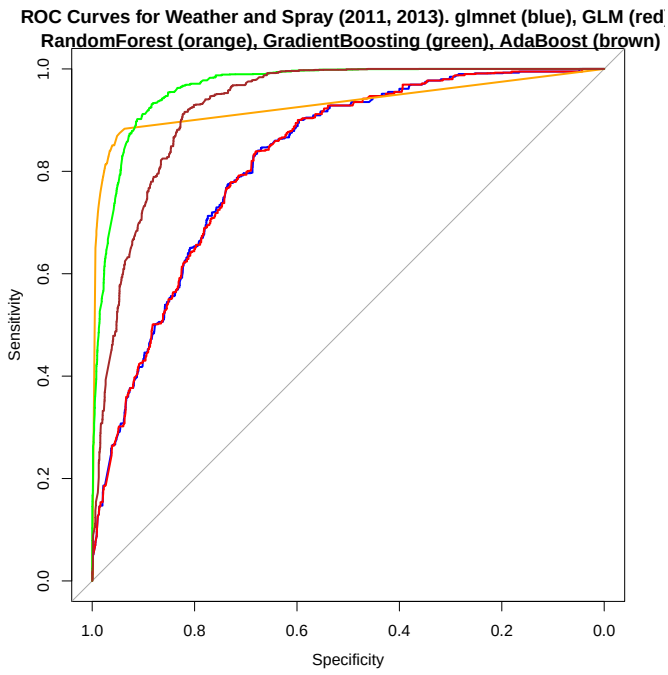
Output of Nonlinear Terms from GAM on Holdout Data



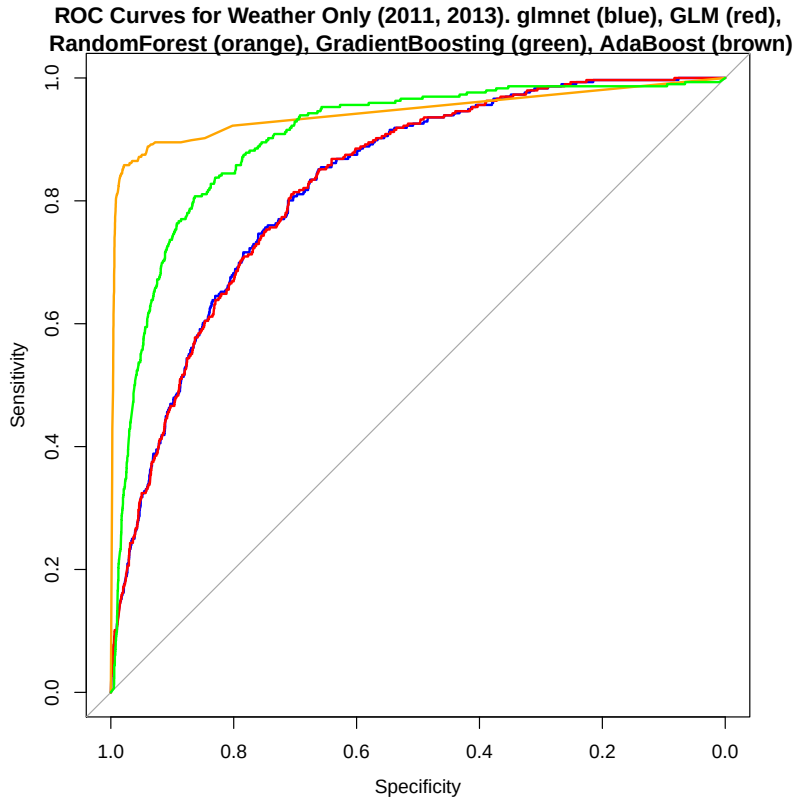
Heatmap of Latitude and Longitude on West Nile Virus incidence



ROC Curve for All Years Data, Weather & Trap Predictors



ROC Curve for Spray Years, Weather, Trap, & Spray Predictors



ROC Curve for All Years Data, Weather & Trap Predictors Only

	Weather AllYears	Weather, SprayYears	Spray & Weather, SprayYears
GLM	0.80	0.82	0.82
GLM Elastic Net	0.80	0.82	0.82
RandomForest	0.85	0.93	0.93
Gradient Boosting	0.93	0.93	0.96
AdaBoost	0.87	0.92	0.93

Optimal AUCs trained on the whole dataset.

5.2 Source Code

All code is available on the personal GitHub repository: `https://github.com/christopheraden/BiostatMasters-Oral`. Code is provided here for posterity.

MakeData:

```
if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
library(TeachingDemos)
library(gdata)
library(data.table)
library(caret)
library(doMC)
library(lubridate)
library(Metrics)
library(geosphere)
library(ggplot2)
library(gam)
library(pROC)
library(parallel)
library(geosphere)

train = fread("data/train.csv")
train$WnvPresent = factor(train$WnvPresent, labels=c("NotPresent", "Present"))
train$Species = factor(train$Species, labels = c("Erraticus", "Pipiens","Pipiens.Restuans", "Restuans", "Salinarius", "Tarsal", "Territans"))
train$Species = as.integer(train$Species)
train$Trap = factor(train$Trap)
train$Year = year(train$Date)
train$Month = month(train$Date)
train$Date = as.Date(train$Date)
train$.outcome = train$WnvPresent

weather = fread("data/weather.csv")
weather[weather=="M"] = NA
weather$Date = as.Date(weather$Date)
weather$Tmax = as.numeric(weather$Tmax)
weather$Tmin = as.numeric(weather$Tmin)
weather$Tavg = as.numeric(weather$Tavg)
weather$Tmax = as.numeric(weather$Tmax)
weather$DewPoint = as.numeric(weather$DewPoint)
weather$WetBulb = as.numeric(weather$WetBulb)
weather$Heat = as.numeric(weather$Heat)
weather$Cool = as.numeric(weather$Cool)
weather$DewPoint = as.numeric(weather$DewPoint)
weather$PrecipTotal[which(weather$PrecipTotal == "  T")] = .005
weather$PrecipTotal = as.numeric(weather$PrecipTotal)
weather$StnPressure = as.numeric(weather$StnPressure)
weather$SeaLevel = as.numeric(weather$SeaLevel)
weather$ResultSpeed = as.numeric(weather$ResultSpeed)
weather$ResultDir = as.numeric(weather$ResultDir)
weather$AvgSpeed = as.numeric(weather$AvgSpeed)
weather$Depart = weather$Sunrise = weather$Sunset = weather$Depth = weather$Water1 = weather$SnowFall = NULL

weatherAgg = as.data.table(aggregate(weather[,!"CodeSum", with=FALSE], by = list(weather$Date), FUN = mean, na.rm=TRUE))
weatherAgg$Date = NULL
WAnames = names(weatherAgg); WAnames[1] = "Date"
setnames(weatherAgg, names(weatherAgg), WAnames)
bigTrain = merge(train, weatherAgg, by="Date")
bigTrain$Address = bigTrain$Street = bigTrain$AddressNumberAndStreet = bigTrain$AddressAccuracy = bigTrain$Station = NULL

#Add in Spraying/Weather Effects:
spray = fread("data/spray.csv")
spray$Date = as.Date(spray$Date)
bigTrain$Date = as.Date(bigTrain$Date)

bigTrain2011_2013 = bigTrain[year(bigTrain$Date) %in% c(2011, 2013), ]
uniqueSprays = unique(spray[, .(Date, Latitude, Longitude)])

```

```
sprayTrapDist = distm(cbind(bigTrain2011_2013$Longitude, bigTrain2011_2013$Latitude),
                      cbind(uniqueSprays$Longitude, uniqueSprays$Latitude),
                      fun = distHaversine) * 0.000621371 #Convert Haversine dist to Mi
whichDists = lapply(1:nrow(bigTrain2011_2013), function(r) which(sprayTrapDist[r,] < 2))
recentlySprayedCloseBy = lapply(1:nrow(bigTrain2011_2013), function(i)
  which(bigTrain2011_2013$Date[i] - uniqueSprays[whichDists[[i]] , Date] < 7 &
          bigTrain2011_2013$Date[i] - uniqueSprays[whichDists[[i]] , Date] >= 0))
bigTrain2011_2013$NearbySprays = sapply(recentlySprayedCloseBy, length)

keep(bigTrain, bigTrain2011_2013, sure = TRUE)
save(bigTrain, file="data/bigTrain.rdata")
save(bigTrain2011_2013, file="data/bigTrain2011_2013.rdata")

bigTrain = bigTrain[rep(seq_len(nrow(bigTrain)), times=bigTrain$NumMosquitos),]
bigTrain2011_2013 = bigTrain2011_2013[rep(seq_len(nrow(bigTrain2011_2013)), times=bigTrain2011_2013$
    NumMosquitos),]
bigTrain$NumMosquitos = bigTrain2011_2013$NumMosquitos = NULL
save(bigTrain, file="data/bigTrain_1MosquitoPerRow.rdata")
save(bigTrain2011_2013, file="data/bigTrain2011_2013_1MosquitoPerRow.rdata")
```

Init_EDA:

```
if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
library(data.table)
library(caret)
library(doMC)
library(lubridate)
library(Metrics)
library(ggmap)
library(vcd)

load("data/bigTrain.rdata")
cdplot(WnvPresent~NumMosquitos, data=bigTrain, main="Conditional Probability of Number of Mosquitos on
    West Nile Presence")

load("data/bigTrain_1MosquitoPerRow.rdata")
mosaic(bigTrain$WnvPresent~bigTrain$Species, main="Mosaic Plot of Mosquito Species vs. WNv Presence")

par(mfrow=1:2)
cdplot(WnvPresent~Latitude, data=bigTrain, main=NULL)
cdplot(WnvPresent~Longitude, data=bigTrain, main=NULL)

#Species of Mosquito on Presence of WNV
with(bigTrain, chisq.test(WnvPresent, Species, simulate.p.value = TRUE, B=1E4)) #An association, not
    surprising.

#Back to better labeling scheme.
bigTrain$Species = factor(bigTrain$Species, labels = c("Erraticus", "Pipiens","Pipiens/Restuans", "
    Restuans", "Salinarius", "Tarsal", "Territans"))

#Data within Traps may be correlated, by nature of trap's location
with(bigTrain, table(WnvPresent, Trap))
chisq.test(bigTrain$WnvPresent, bigTrain$Trap, simulate.p.value = TRUE, B=1E4)
#Could use as poor man's location effect?
TrapLogit = glm(WnvPresent ~ Trap + Species, data=bigTrain, family=binomial)
summary(TrapLogit)
```

Inference:

```
if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
library(car)
library(MASS)
library(pROC)
library(glmnet)
library(doMC)
library(parallel)
library(mgcv)
library(LogisticDx)
library(genridge)
library(geepack)

```

```
#2011 & 2013 years, with Sprays.
load("data/bigTrain2011_2013_1MosquitoPerRow.rdata")
train = bigTrain2011_2013; rm(bigTrain2011_2013)
train$WnvPresent = train$.outcome; train$.outcome = NULL
n = nrow(train)
n_eff = sample(1:n, .8*n)
holdout = setdiff(1:n, n_eff)

par(mfrow=1:2)
truehist(train$NearbySprays, xlab="Number of Nearby Sprays", main="Histogram of Nearby Sprays", prob = FALSE)
truehist(train$NearbySprays[train$NearbySprays>0], xlab="Number of Nearby Nonzero Sprays", main="Histogram of Nearby Nonzero Sprays", prob=FALSE)
spineplot(WnvPresent~NearbySprays, xlim=c(0, 1), data=train, main="", xlab="Nearby Sprays")
cdplot(WnvPresent~NearbySprays, data=train[train$NearbySprays>0,], main="", xlab="Nearby Sprays > 0")

SprayEffectCts = glm(WnvPresent ~ . -Species -Trap -Block -Tmax -Tmin -Cool -Date + as.factor(Month) - Month + as.factor(Year) -Year + Latitude * Longitude + as.factor(Species) + NearbySprays, data=train, family='binomial')
SprayEffectDich =glm(WnvPresent ~ . -Species -Trap -Block -Tmax -Tmin -Cool -Date + as.factor(Month) - Month + as.factor(Year) -Year + Latitude * Longitude + as.factor(Species) -NearbySprays + as.factor(NearbySprays>0),data=train, family='binomial')
summary(SprayEffectCts)
summary(SprayEffectDich)

foo = formula((as.numeric(WnvPresent)-1) ~ . - Species - Trap - Block - Tmax - Tmin - Cool -
  Date + as.factor(Month) - Month + as.factor(Year) - Year +
  Latitude * Longitude + as.factor(Species) + as.factor(NearbySprays > 0) -NearbySprays)

ridgeGCV = function(Lambda, X=train[n_eff,]){
  log(genridge::ridge(as.numeric(WnvPresent)-1 ~ . - Species - Trap - Block - Tmax - Tmin - Cool -
              Date + as.factor(Month) - Month + as.factor(Year) - Year +
              Latitude * Longitude + as.factor(Species) + as.factor(NearbySprays > 0) -NearbySprays,
            data=X, lambda=Lambda, family='binomial')$GCV)
}
lambda_opt = optimize(f = ridgeGCV, lower=0, upper=1E4)$minimum

SprayEffectCtsNoRidge = genridge::ridge(as.numeric(WnvPresent)-1 ~ . - Species - Trap - Block - Tmax - Tmin - Cool -
                                Date + as.factor(Month) - Month + as.factor(Year) - Year +
                                Latitude * Longitude + as.factor(Species) + NearbySprays,
                              data=train[n_eff,], lambda=0, family='binomial')
SprayEffectDichNoRidge = genridge::ridge(as.numeric(WnvPresent)-1 ~ . - Species - Trap - Block - Tmax - Tmin - Cool -
                                  Date + as.factor(Month) - Month + as.factor(Year) - Year +
                                  Latitude * Longitude + as.factor(Species) -NearbySprays + as.factor(NearbySprays > 0),
                                data=train[n_eff,], lambda=0, family='binomial')
SprayEffectDichRidge = genridge::ridge(as.numeric(WnvPresent)-1 ~ . - Species - Trap - Block - Tmax - Tmin - Cool -
                                Date + as.factor(Month) - Month + as.factor(Year) - Year +
                                Latitude * Longitude + as.factor(Species) + as.factor(NearbySprays > 0) - NearbySprays,
                              data=train[n_eff,], lambda=lambda_opt, family='binomial')

VIFs = data.frame(t(vif.ridge(SprayEffectCtsNoRidge)),
                  t(vif.ridge(SprayEffectDichNoRidge)),
                  t(vif.ridge(SprayEffectDichRidge)))
colnames(VIFs) = c("NoRidgeContinuous", "NoRidgeDiscrete", "RidgingDiscrete(lambda=3.596)")

train$Month = as.factor(train$Month)
train$Year = as.factor(train$Year)
train$Species = as.factor(train$Species)

X.design = model.matrix(~ . -WnvPresent -Trap -Cool -Block -Date + Latitude * Longitude -NearbySprays + as.factor(NearbySprays > 0)-1, data=train[n_eff,])
Y.response = as.numeric(train$WnvPresent[n_eff])-1

glmNoReg = glm(WnvPresent~.-Trap-Cool -Block -Date -Tmax - Tmin + Latitude * Longitude -NearbySprays + as.factor(NearbySprays > 0), data=train[n_eff,], family="binomial")
summary(glmNoReg)
vif(glmNoReg)

registerDoMC(detectCores())

CV = cv.glmnet(X.design, Y.response, family='binomial', alpha=.9, parallel = TRUE)
```

```
glmnetFit = glmnet(x = X.design, y = Y.response, lambda = CV$lambda.1se, family='binomial', alpha = .9)
coef(glmnetFit, CV$lambda.1se) #Best model, chosen by Lasso

train$isSpecies1247 = factor(train$Species %in% c("1", "2", "4", "7"))
train$anyNearbySprays = factor(train$NearbySprays > 0)

glmHoldout = glm(WnvPresent ~ isSpecies1247 + Latitude*Longitude-Latitude + Year + Month + Tmax + Heat +
    StnPressure + ResultSpeed + anyNearbySprays, data = train[holdout,], family='binomial')
summary(glmHoldout)
vif(glmHoldout)

gamHoldout = gam(WnvPresent ~ isSpecies1247 + te(Longitude, Latitude) + Year + Month + s(Tmax) + Heat +s(
    StnPressure) +s(ResultSpeed) +anyNearbySprays, data = train[holdout,], family='binomial')
summary(gamHoldout)
par(mfrow=c(1,1))
plot(gamHoldout, select = 1, scheme = 2, main="Heatmap of Longitude and Latitude vs. WNv Incidence")

glmHoldout_RandomEffect <- geese(as.numeric(WnvPresent)-1 ~ isSpecies1247 + ns(Longitude, 3) + Year +
    Month + Tmax + Heat + StnPressure + ResultSpeed + anyNearbySprays,
                        id = Trap, data = train[holdout,], family = binomial, corstr="exchangeable", scale
                            .fix= TRUE)
summary(glmHoldout_RandomEffect)
```

WeatherModel_Prediction:

```
if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
library(TeachingDemos)
library(gdata)
library(data.table)
library(caret)
library(doMC)
library(lubridate)
library(Metrics)
library(geosphere)
library(ggplot2)
library(gam)
library(pROC)
library(parallel)

load("data/bigTrain_1MosquitoPerRow.rdata")
bigTrain$WnvPresent = bigTrain$Date = bigTrain$NumMosquitos = bigTrain$Trap = NULL

twoClassSummaryCustom = function(data, lev = NULL, model = NULL) {
  rocObject <- try(pROC::roc(data$obs, data[, lev[1]]), silent = TRUE)
  rocAUC <- ifelse(class(rocObject)[1] == "try-error", NA, rocObject$auc)
  sens = sensitivity(data[, "pred"], data[, "obs"], lev[1])
  spec = specificity(data[, "pred"], data[, "obs"], lev[2])
  ppv = posPredValue(data[,"pred"], reference = data[,"obs"])
  BalancedAccuracy = mean(c(sens, spec))
  F1 = 2 * (ppv * sens) / (ppv+sens)
  out <- c(F1, BalancedAccuracy, rocAUC, sens, spec)
  names(out) <- c("F1", "BalancedAccuracy", "AUC", "Sens", "Spec")
  out
}

CVctrl = trainControl(method='cv', number=5, classProbs=TRUE, summaryFunction=twoClassSummaryCustom)
registerDoMC(detectCores())

WeatherModel = formula(.outcome ~ .)
WeatherModelLM = formula(.outcome ~ . + as.factor(Species) + as.factor(Year) + as.factor(Month) -Month -
    Cool -Species -Year)

fitGLM = train(WeatherModelLM, data = bigTrain, trControl=CVctrl, metric="F1", method = "glm", family='
    binomial')

GLMNet_Grid = expand.grid(alpha=c(seq(0, .1, .001), seq(.9, 1,.001)), lambda=seq(0,.1,.001))
fitGLMNet_F1 = train(WeatherModelLM, data = bigTrain, method = "glmnet", trControl = CVctrl, metric = 'F1'
    , preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
fitGLMNet_BA = train(WeatherModelLM, data = bigTrain, method = "glmnet", trControl = CVctrl, metric = '
    BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
fitGLMNet_AUC = train(WeatherModelLM, data = bigTrain, method = "glmnet", trControl = CVctrl, metric = '
    AUC', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)

RF_Grid = expand.grid(mtry=seq(1, 30, 2))
```

```
fitRF_F1 = train(WeatherModel, data = bigTrain, method = "rf",  trControl = CVctrl, metric = 'F1', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
fitRF_BA = train(WeatherModel, data = bigTrain, method = "rf",  trControl = CVctrl, metric = 'BalancedAccuracy', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
fitRF_AUC = train(WeatherModel, data = bigTrain, method = "rf",  trControl = CVctrl, metric = 'AUC', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)

GBM_Grid = expand.grid(n.trees = seq(1, 300, 50), interaction.depth = 2:3, shrinkage=seq(0, 1, .1), n.minobsinnode = seq(10, 50, 10))
fitGBM_F1 = train(WeatherModel, data = bigTrain, method = "gbm", trControl = CVctrl, metric = 'F1', preProc=c("center","scale"), verbose=FALSE)
fitGBM_BA = train(WeatherModel, data = bigTrain, method = "gbm", trControl = CVctrl, metric = 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)
fitGBM_AUC = train(WeatherModel, data = bigTrain, method = "gbm", trControl = CVctrl, metric = 'AUC', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)

Ada_Grid = expand.grid(iter=150, maxdepth=1:3, nu=seq(0, 1, .1))
fitAda_F1 = train(WeatherModel, data = bigTrain, method = "ada", trControl = CVctrl, metric= 'F1', preProc=c("center","scale"), tuneGrid = Ada_Grid)
fitAda_BA = train(WeatherModel, data = bigTrain, method = "ada", trControl = CVctrl, metric= 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = Ada_Grid)
fitAda_AUC = train(WeatherModel, data = bigTrain, method = "ada", trControl = CVctrl, metric= 'AUC', preProc=c("center","scale"), tuneGrid = Ada_Grid)
save(list = ls()[grep("fit", ls())], file="out/AllFits.Rdata")
```

Spraying_Prediction:

```
if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
library(TeachingDemos)
library(gdata)
library(data.table)
library(caret)
library(doMC)
library(lubridate)
library(Metrics)
library(geosphere)
library(ggplot2)
library(gam)
library(pROC)
library(parallel)

load("data/bigTrain2011_2013_1MosquitoPerRow.rdata")
bigTrain2011_2013$WnvPresent = bigTrain2011_2013$NumMosquitos = bigTrain2011_2013$Date = bigTrain2011_2013$Trap = NULL

twoClassSummaryCustom = function(data, lev = NULL, model = NULL) {
  rocObject <- try(pROC::roc(data$obs, data[, lev[1]]), silent = TRUE)
  rocAUC <- ifelse(class(rocObject)[1] == "try-error", NA, rocObject$auc)
  sens = sensitivity(data[, "pred"], data[, "obs"], lev[1])
  spec = specificity(data[, "pred"], data[, "obs"], lev[2])
  ppv = posPredValue(data[,"pred"], reference = data[,"obs"])
  BalancedAccuracy = mean(c(sens, spec))
  F1 = 2 * (ppv * sens) / (ppv+sens)
  out <- c(F1, BalancedAccuracy, rocAUC, sens, spec)
  names(out) <- c("F1", "BalancedAccuracy", "AUC", "Sens", "Spec")
  out
}

CVctrl = trainControl(method='cv', number=5, classProbs=TRUE, summaryFunction=twoClassSummaryCustom)
registerDoMC(detectCores())

SprayModel = formula(.outcome ~ .)
SprayModelLM = formula(.outcome ~ . + as.factor(Species) + as.factor(Year) + as.factor(Month) + as.factor(NearbySprays>0) -Month -Cool -Species -Year -NearbySprays)

fitGLM = train(SprayModelLM, data = bigTrain2011_2013, trControl=CVctrl, metric="AUC", method = "glm", family='binomial')

GLMNet_Grid = expand.grid(alpha=c(seq(0, .1, .001), seq(.9, 1,.001)), lambda=seq(0,.1,.001))
fitGLMNet_F1 = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric = 'F1', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
fitGLMNet_BA = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric = 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
fitGLMNet_AUC = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric = 'AUC', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
```

```
43
44 RF_Grid = expand.grid(mtry=seq(1, 30, 2))
45 fitRF_F1 = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = 'F1',
       importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
46 fitRF_BA = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = '
       BalancedAccuracy', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
47 fitRF_AUC = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = 'AUC'
       , importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
48
49 GBM_Grid = expand.grid(n.trees = seq(1, 300, 50), interaction.depth = 2:3, shrinkage=seq(0, 1, .1), n.
       minobsinnode = seq(10, 50, 10))
50 fitGBM_F1 = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = 'F1',
        preProc=c("center","scale"), verbose=FALSE)
51 fitGBM_BA = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = '
       BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)
52 fitGBM_AUC = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = 'AUC
       ', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)
53
54 Ada_Grid = expand.grid(iter=150, maxdepth=1:3, nu=seq(0, 1, .1))
55 fitAda_F1 = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= 'F1',
       preProc=c("center","scale"), tuneGrid = Ada_Grid)
56 fitAda_BA = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= '
       BalancedAccuracy', preProc=c("center","scale"), tuneGrid = Ada_Grid)
57 fitAda_AUC = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= 'AUC'
       , preProc=c("center","scale"), tuneGrid = Ada_Grid)
58 save(list = ls()[grep("fit", ls())], file="out/SprayFits.Rdata")
```

Weather_Prediction_Restricted:

```
1  if(Sys.info()['user']=="Aden") { setwd("~/Dropbox/School/MS Exam/") } else setwd("~/MS\ Exam/")
2  library(TeachingDemos)
3  library(gdata)
4  library(data.table)
5  library(caret)
6  library(doMC)
7  library(lubridate)
8  library(Metrics)
9  library(geosphere)
10 library(ggplot2)
11 library(gam)
12 library(pROC)
13 library(parallel)
14
15 load("data/bigTrain2011_2013_1MosquitoPerRow.rdata")
16 bigTrain2011_2013$WnvPresent = bigTrain2011_2013$Date = bigTrain2011_2013$NumMosquitos = bigTrain2011_2013
       $NearbySprays = bigTrain2011_2013$Trap = NULL
17
18 twoClassSummaryCustom = function(data, lev = NULL, model = NULL) {
19   rocObject <- try(pROC::roc(data$obs, data[, lev[1]]), silent = TRUE)
20   rocAUC <- ifelse(class(rocObject)[1] == "try-error", NA, rocObject$auc)
21   sens = sensitivity(data[, "pred"], data[, "obs"], lev[1])
22   spec = specificity(data[, "pred"], data[, "obs"], lev[2])
23   ppv = posPredValue(data[,"pred"], reference = data[,"obs"])
24   BalancedAccuracy = mean(c(sens, spec))
25   F1 = 2 * (ppv * sens) / (ppv+sens)
26   out <- c(F1, BalancedAccuracy, rocAUC, sens, spec)
27   names(out) <- c("F1", "BalancedAccuracy", "AUC", "Sens", "Spec")
28   out
29 }
30
31 CVctrl = trainControl(method='cv', number=5, classProbs=TRUE, summaryFunction=twoClassSummaryCustom)
32 registerDoMC(detectCores())
33
34 SprayModel = formula(.outcome ~ .)
35 SprayModelLM = formula(.outcome ~ . + as.factor(Species) + as.factor(Year) + as.factor(Month) -Month -Cool
       -Species -Year)
36
37 fitGLM = train(SprayModelLM, data = bigTrain2011_2013, trControl=CVctrl, metric="AUC", method = "glm",
       family='binomial')
38
39 GLMNet_Grid = expand.grid(alpha=c(seq(0, .1, .001), seq(.9, 1,.001)), lambda=seq(0,.1,.001))
40 fitGLMNet_F1 = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric
       = 'F1', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
41 fitGLMNet_BA = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric
       = 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)
```

```
fitGLMNet_AUC = train(SprayModelLM, data = bigTrain2011_2013, method = "glmnet", trControl = CVctrl, metric = 'AUC', preProc=c("center","scale"), tuneGrid = GLMNet_Grid)

RF_Grid = expand.grid(mtry=seq(1, 30, 2))
fitRF_F1 = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = 'F1', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
fitRF_BA = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = 'BalancedAccuracy', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)
fitRF_AUC = train(SprayModel, data = bigTrain2011_2013, method = "rf", trControl = CVctrl, metric = 'AUC', importance=TRUE, preProc=c("center","scale"), tuneGrid = RF_Grid)

GBM_Grid = expand.grid(n.trees = seq(1, 300, 50), interaction.depth = 2:3, shrinkage=seq(0, 1, .1), n.minobsinnode = seq(10, 50, 10))
fitGBM_F1 = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = 'F1', preProc=c("center","scale"), verbose=FALSE)
fitGBM_BA = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)
fitGBM_AUC = train(SprayModel, data = bigTrain2011_2013, method = "gbm", trControl = CVctrl, metric = 'AUC', preProc=c("center","scale"), tuneGrid = GBM_Grid, verbose=FALSE)

Ada_Grid = expand.grid(iter=150, maxdepth=1:3, nu=seq(0, 1, .1))
fitAda_F1 = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= 'F1', preProc=c("center","scale"), tuneGrid = Ada_Grid)
fitAda_BA = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= 'BalancedAccuracy', preProc=c("center","scale"), tuneGrid = Ada_Grid)
fitAda_AUC = train(SprayModel, data = bigTrain2011_2013, method = "ada", trControl = CVctrl, metric= 'AUC', preProc=c("center","scale"), tuneGrid = Ada_Grid)
save(list = ls()[grep("fit", ls())], file="SprayFits_NoSpray.Rdata")
```

ROC_PredictionResults:

```
library(pROC)
library(TeachingDemos)
library(caret)

load("data/bigTrain_1MosquitoPerRow.rdata")
load("data/bigTrain2011_2013_1MosquitoPerRow.rdata")

load("out/AllFits.Rdata")
pdf(file="out/ROC_AllYears_Weather.pdf")
plot(roc(bigTrain$.outcome ~ predict(fitGLMNet_AUC, bigTrain, type='prob')$Present), col='blue', xlim=c(1,0), main="ROC Curves for Weather Only (2007-2013). glmnet (blue), GLM (red),\n RandomForest (orange), GradientBoosting (green), AdaBoost (brown)")
plot(roc(bigTrain$.outcome ~ predict(fitGLM, bigTrain, type='prob')$Present), col='red', add=TRUE) #GLM ROC
plot(roc(bigTrain$.outcome ~ predict(fitRF_AUC, bigTrain, type='prob')$Present), col='orange', add=TRUE) #RandomForest ROC
plot(roc(bigTrain$.outcome ~ predict(fitGBM_AUC, bigTrain, type='prob')$Present), col='green', add=TRUE) #GBM ROC
plot(roc(bigTrain$.outcome ~ predict(fitAda_AUC, bigTrain, type = "prob")$Present), col='brown', add=TRUE) #Ada ROC
dev.off()

fitAllResults = data.frame(BestF1 = c(fitGLM[[4]]$F1[which.max(fitGLM[[4]]$F1)],
                                      fitGLMNet_F1[[4]]$F1[which.max(fitGLMNet_F1[[4]]$F1)],
                                      fitRF_F1[[4]]$F1[which.max(fitRF_F1[[4]]$F1)],
                                      fitGBM_F1[[4]]$F1[which.max(fitGBM_F1[[4]]$F1)],
                                      fitAda_F1[[4]]$F1[which.max(fitAda_F1[[4]]$F1)]),
                           BestF1SD = c(fitGLM[[4]]$F1SD[which.max(fitGLM[[4]]$F1SD)],
                                        fitGLMNet_F1[[4]]$F1SD[which.max(fitGLMNet_F1[[4]]$F1SD)],
                                        fitRF_F1[[4]]$F1SD[which.max(fitRF_F1[[4]]$F1SD)],
                                        fitGBM_F1[[4]]$F1SD[which.max(fitGBM_F1[[4]]$F1SD)],
                                        fitAda_F1[[4]]$F1SD[which.max(fitAda_F1[[4]]$F1SD)]),
                           BestAUC = c(fitGLM[[4]]$AUC[which.max(fitGLM[[4]]$AUC)],
                                       fitGLMNet_AUC[[4]]$AUC[which.max(fitGLMNet_AUC[[4]]$AUC)],
                                       fitRF_AUC[[4]]$AUC[which.max(fitRF_AUC[[4]]$AUC)],
                                       fitGBM_AUC[[4]]$AUC[which.max(fitGBM_AUC[[4]]$AUC)],
                                       fitAda_AUC[[4]]$AUC[which.max(fitAda_AUC[[4]]$AUC)]),
                           BestAUCSD = c(fitGLM[[4]]$AUCSD[which.max(fitGLM[[4]]$AUCSD)],
                                         fitGLMNet_AUC[[4]]$AUCSD[which.max(fitGLMNet_AUC[[4]]$AUCSD)],
                                         fitRF_AUC[[4]]$AUCSD[which.max(fitRF_AUC[[4]]$AUCSD)],
                                         fitGBM_AUC[[4]]$AUCSD[which.max(fitGBM_AUC[[4]]$AUCSD)],
                                         fitAda_AUC[[4]]$AUCSD[which.max(fitAda_AUC[[4]]$AUCSD)]),
                           BestBalancedAccuracy = c(fitGLM[[4]]$BalancedAccuracy[which.max(fitGLM[[4]]$
                                BalancedAccuracy)],
```

```
                                                fitGLMNet_BA[[4]]$BalancedAccuracy[which.max(fitGLMNet
                                                    _BA[[4]]$BalancedAccuracy)],
                                                fitRF_BA[[4]]$BalancedAccuracy[which.max(fitRF_BA[[4]]
                                                    $BalancedAccuracy)],
                                                fitGBM_BA[[4]]$BalancedAccuracy[which.max(fitGBM_BA
                                                    [[4]]$BalancedAccuracy)],
                                                fitAda_BA[[4]]$BalancedAccuracy[which.max(fitAda_BA
                                                    [[4]]$BalancedAccuracy)]),
                                BestBalancedAccuracySD = c(fitGLM[[4]]$BalancedAccuracySD[which.max(fitGLM[[4]]
                                    $BalancedAccuracySD)],
                                                  fitGLMNet_BA[[4]]$BalancedAccuracySD[which.max(
                                                      fitGLMNet_BA[[4]]$BalancedAccuracySD)],
                                                  fitRF_BA[[4]]$BalancedAccuracySD[which.max(fitRF_BA
                                                      [[4]]$BalancedAccuracySD)],
                                                  fitGBM_BA[[4]]$BalancedAccuracySD[which.max(fitGBM_
                                                      BA[[4]]$BalancedAccuracySD)],
                                                  fitAda_BA[[4]]$BalancedAccuracySD[which.max(fitAda_
                                                      BA[[4]]$BalancedAccuracySD)]))
rownames(fitAllResults) = c("GLM", "GLM Elastic Net", "RandomForest", "Gradient Boosting", "AdaBoost")

txtStart(file = "out/AllTrain_Confusion.txt")
confusionMatrix(fitGLM)
confusionMatrix(fitGLMNet_F1); confusionMatrix(fitGLMNet_BA); confusionMatrix(fitGLMNet_AUC)
confusionMatrix(fitRF_F1); confusionMatrix(fitRF_BA); confusionMatrix(fitRF_AUC)
confusionMatrix(fitGBM_F1); confusionMatrix(fitGBM_BA); confusionMatrix(fitGBM_AUC)
confusionMatrix(fitAda_F1); confusionMatrix(fitAda_BA); confusionMatrix(fitAda_AUC)
txtStop()









load("out/SprayFits_NoSpray.Rdata")
pdf(file="out/ROC_SprayYears_WeatherOnly.pdf")
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGLMNet_AUC, bigTrain2011_2013, type='prob')$Present), col
    ='blue', xlim=c(1,0), main="ROC Curves for Weather Only (2011, 2013). glmnet (blue), GLM (red),\n
    RandomForest (orange), GradientBoosting (green), AdaBoost (brown)")
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGLM, bigTrain2011_2013, type='prob')$Present), col='red',
     add=TRUE) #GLM ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitRF_AUC, bigTrain2011_2013, type='prob')$Present), col='
    orange', add=TRUE) #RandomForest ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGBM_AUC, bigTrain2011_2013, type='prob')$Present), col='
    green', add=TRUE) #GBM ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitAda_AUC, bigTrain2011_2013, type = "prob")$Present), col=
    'brown', add=TRUE) #Ada ROC
dev.off()

fitSprayResults_NoSpray = data.frame(BestF1 = c(fitGLM[[4]]$F1[which.max(fitGLM[[4]]$F1)],
                                                fitGLMNet_F1[[4]]$F1[which.max(fitGLMNet_F1[[4]]$F1)],
                                                fitRF_F1[[4]]$F1[which.max(fitRF_F1[[4]]$F1)],
                                                fitGBM_F1[[4]]$F1[which.max(fitGBM_F1[[4]]$F1)],
                                                fitAda_F1[[4]]$F1[which.max(fitAda_F1[[4]]$F1)]),
                                     BestF1SD = c(fitGLM[[4]]$F1SD[which.max(fitGLM[[4]]$F1SD)],
                                                  fitGLMNet_F1[[4]]$F1SD[which.max(fitGLMNet_F1[[4]]$F1SD)
                                                      ],
                                                  fitRF_F1[[4]]$F1SD[which.max(fitRF_F1[[4]]$F1SD)],
                                                  fitGBM_F1[[4]]$F1SD[which.max(fitGBM_F1[[4]]$F1SD)],
                                                  fitAda_F1[[4]]$F1SD[which.max(fitAda_F1[[4]]$F1SD)]),
                                     BestAUC = c(fitGLM[[4]]$AUC[which.max(fitGLM[[4]]$AUC)],
                                                 fitGLMNet_AUC[[4]]$AUC[which.max(fitGLMNet_AUC[[4]]$AUC)
                                                     ],
                                                 fitRF_AUC[[4]]$AUC[which.max(fitRF_AUC[[4]]$AUC)],
                                                 fitGBM_AUC[[4]]$AUC[which.max(fitGBM_AUC[[4]]$AUC)],
                                                 fitAda_AUC[[4]]$AUC[which.max(fitAda_AUC[[4]]$AUC)]),
                                     BestAUCSD = c(fitGLM[[4]]$AUCSD[which.max(fitGLM[[4]]$AUCSD)],
                                                   fitGLMNet_AUC[[4]]$AUCSD[which.max(fitGLMNet_AUC[[4]]$
                                                       AUCSD)],
                                                   fitRF_AUC[[4]]$AUCSD[which.max(fitRF_AUC[[4]]$AUCSD)],
                                                   fitGBM_AUC[[4]]$AUCSD[which.max(fitGBM_AUC[[4]]$AUCSD)
                                                       ],
                                                   fitAda_AUC[[4]]$AUCSD[which.max(fitAda_AUC[[4]]$AUCSD)
                                                       ]),
                                     BestBalancedAccuracy = c(fitGLM[[4]]$BalancedAccuracy[which.max(
                                         fitGLM[[4]]$BalancedAccuracy)],
                                                              fitGLMNet_BA[[4]]$BalancedAccuracy[which.max
                                                                  (fitGLMNet_BA[[4]]$BalancedAccuracy)],
                                                              fitRF_BA[[4]]$BalancedAccuracy[which.max(
```

```
                                                    fitRF_BA[[4]]$BalancedAccuracy)],
                                                  fitGBM_BA[[4]]$BalancedAccuracy[which.max(
                                                    fitGBM_BA[[4]]$BalancedAccuracy)],
                                                  fitAda_BA[[4]]$BalancedAccuracy[which.max(
                                                    fitAda_BA[[4]]$BalancedAccuracy)]),
                                BestBalancedAccuracySD = c(fitGLM[[4]]$BalancedAccuracySD[which.max(
                                    fitGLM[[4]]$BalancedAccuracySD)],
                                                  fitGLMNet_BA[[4]]$BalancedAccuracySD[which
                                                    .max(fitGLMNet_BA[[4]]$
                                                    BalancedAccuracySD)],
                                                  fitRF_BA[[4]]$BalancedAccuracySD[which.max
                                                    (fitRF_BA[[4]]$BalancedAccuracySD)],
                                                  fitGBM_BA[[4]]$BalancedAccuracySD[which.
                                                    max(fitGBM_BA[[4]]$BalancedAccuracySD)
                                                    ],
                                                  fitAda_BA[[4]]$BalancedAccuracySD[which.
                                                    max(fitAda_BA[[4]]$BalancedAccuracySD)
                                                    ]))
rownames(fitSprayResults_NoSpray) = c("GLM", "GLM Elastic Net", "RandomForest", "Gradient Boosting", "
    AdaBoost")

txtStart(file = "SprayTrain_NoSpray_Confusion.txt")
confusionMatrix(fitGLM)
confusionMatrix(fitGLMNet_F1); confusionMatrix(fitGLMNet_BA); confusionMatrix(fitGLMNet_AUC)
confusionMatrix(fitRF_F1); confusionMatrix(fitRF_BA); confusionMatrix(fitRF_AUC)
confusionMatrix(fitGBM_F1); confusionMatrix(fitGBM_BA); confusionMatrix(fitGBM_AUC)
confusionMatrix(fitAda_F1); confusionMatrix(fitAda_BA); confusionMatrix(fitAda_AUC)
txtStop()



load("out/SprayFits.Rdata")
pdf(file="out/ROC_SprayYears_SprayWeather.pdf")
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGLMNet_AUC, bigTrain2011_2013, type='prob')$Present), col
    ='blue', xlim=c(1,0), main="ROC Curves for Weather and Spray (2011, 2013). glmnet (blue), GLM (red),\n
     RandomForest (orange), GradientBoosting (green), AdaBoost (brown)")
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGLM, bigTrain2011_2013, type='prob')$Present), col='red',
     add=TRUE) #GLM ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitRF_AUC, bigTrain2011_2013, type='prob')$Present), col='
    orange', add=TRUE) #RandomForest ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitGBM_AUC, bigTrain2011_2013, type='prob')$Present), col='
    green', add=TRUE) #GBM ROC
plot(roc(bigTrain2011_2013$.outcome ~ predict(fitAda_AUC, bigTrain2011_2013, type = "prob")$Present), col=
    'brown', add=TRUE) #Ada ROC
dev.off()

fitSprayResults = data.frame(BestF1 = c(fitGLM[[4]]$F1[which.max(fitGLM[[4]]$F1)],
                                        fitGLMNet_F1[[4]]$F1[which.max(fitGLMNet_F1[[4]]$F1)],
                                        fitRF_F1[[4]]$F1[which.max(fitRF_F1[[4]]$F1)],
                                        fitGBM_F1[[4]]$F1[which.max(fitGBM_F1[[4]]$F1)],
                                        fitAda_F1[[4]]$F1[which.max(fitAda_F1[[4]]$F1)]),
                             BestF1SD = c(fitGLM[[4]]$F1SD[which.max(fitGLM[[4]]$F1SD)],
                                          fitGLMNet_F1[[4]]$F1SD[which.max(fitGLMNet_F1[[4]]$F1SD)],
                                          fitRF_F1[[4]]$F1SD[which.max(fitRF_F1[[4]]$F1SD)],
                                          fitGBM_F1[[4]]$F1SD[which.max(fitGBM_F1[[4]]$F1SD)],
                                          fitAda_F1[[4]]$F1SD[which.max(fitAda_F1[[4]]$F1SD)]),
                             BestAUC = c(fitGLM[[4]]$AUC[which.max(fitGLM[[4]]$AUC)],
                                         fitGLMNet_AUC[[4]]$AUC[which.max(fitGLMNet_AUC[[4]]$AUC)],
                                         fitRF_AUC[[4]]$AUC[which.max(fitRF_AUC[[4]]$AUC)],
                                         fitGBM_AUC[[4]]$AUC[which.max(fitGBM_AUC[[4]]$AUC)],
                                         fitAda_AUC[[4]]$AUC[which.max(fitAda_AUC[[4]]$AUC)]),
                             BestAUCSD = c(fitGLM[[4]]$AUCSD[which.max(fitGLM[[4]]$AUCSD)],
                                           fitGLMNet_AUC[[4]]$AUCSD[which.max(fitGLMNet_AUC[[4]]$AUCSD)],
                                           fitRF_AUC[[4]]$AUCSD[which.max(fitRF_AUC[[4]]$AUCSD)],
                                           fitGBM_AUC[[4]]$AUCSD[which.max(fitGBM_AUC[[4]]$AUCSD)],
                                           fitAda_AUC[[4]]$AUCSD[which.max(fitAda_AUC[[4]]$AUCSD)]),
                             BestBalancedAccuracy = c(fitGLM[[4]]$BalancedAccuracy[which.max(fitGLM[[4]]$
                                 BalancedAccuracy)],
                                                      fitGLMNet_BA[[4]]$BalancedAccuracy[which.max(
                                                          fitGLMNet_BA[[4]]$BalancedAccuracy)],
                                                      fitRF_BA[[4]]$BalancedAccuracy[which.max(fitRF_BA
                                                          [[4]]$BalancedAccuracy)],
                                                      fitGBM_BA[[4]]$BalancedAccuracy[which.max(fitGBM_BA
                                                          [[4]]$BalancedAccuracy)],
                                                      fitAda_BA[[4]]$BalancedAccuracy[which.max(fitAda_BA
                                                          [[4]]$BalancedAccuracy)]),
                             BestBalancedAccuracySD = c(fitGLM[[4]]$BalancedAccuracySD[which.max(fitGLM
                                 [[4]]$BalancedAccuracySD)],
```

```
                                                    fitGLMNet_BA[[4]]$BalancedAccuracySD[which.max(
                                                        fitGLMNet_BA[[4]]$BalancedAccuracySD)],
                                                    fitRF_BA[[4]]$BalancedAccuracySD[which.max(fitRF_
                                                        BA[[4]]$BalancedAccuracySD)],
                                                    fitGBM_BA[[4]]$BalancedAccuracySD[which.max(fitGBM
                                                        _BA[[4]]$BalancedAccuracySD)],
                                                    fitAda_BA[[4]]$BalancedAccuracySD[which.max(fitAda
                                                        _BA[[4]]$BalancedAccuracySD)]))
rownames(fitSprayResults) = c("GLM", "GLM Elastic Net", "RandomForest", "Gradient Boosting", "AdaBoost")

txtStart(file = "out/SprayTrain_Confusion.txt")
confusionMatrix(fitGLM)
confusionMatrix(fitGLMNet_F1); confusionMatrix(fitGLMNet_BA); confusionMatrix(fitGLMNet_AUC)
confusionMatrix(fitRF_F1); confusionMatrix(fitRF_BA); confusionMatrix(fitRF_AUC)
confusionMatrix(fitGBM_F1); confusionMatrix(fitGBM_BA); confusionMatrix(fitGBM_AUC)
confusionMatrix(fitAda_F1); confusionMatrix(fitAda_BA); confusionMatrix(fitAda_AUC)
txtStop()

BestAUC_NoCV = data.frame(Weather_AllYears= Weather_AllYears_AUC,
                          Weather_SprayYears = Weather_SprayYears_AUC,
                          SprayWeather_SprayYears = SprayWeather_SprayYears_AUC)
BestAUC = data.frame(Weather_AllYears=fitAllResults$BestAUC,
                     SprayWeather_SprayYears = fitSprayResults_NoSpray$BestAUC,
                     Weather_SprayYears=fitSprayResults$BestAUC)
BestBA = data.frame(Weather_AllYears=fitAllResults$BestBalancedAccuracy,
                     SprayWeather_SprayYears = fitSprayResults_NoSpray$BestBalancedAccuracy,
                     Weather_SprayYears=fitSprayResults$BestBalancedAccuracy)
rownames(BestAUC_NoCV) = rownames(BestAUC) = rownames(BestBA) = c("GLM", "GLM ElasticNet", "RandomForest",
      "Gradient Boosting", "AdaBoost")


print(xtable(round(BestAUC_NoCV, 3)))
print(xtable(round(BestAUC, 3)))
print(xtable(round(BestBA, 3)))

#Obtained from the plots above.
Weather_AllYears_AUC = c(0.7961, 0.7965, 0.8547, 0.9329, 0.8741)
Weather_SprayYears_AUC = c(0.8195, 0.8196, 0.9267, 0.9271, 0.9234)
SprayWeather_SprayYears_AUC = c(0.8215, 0.8216, 0.9307, 0.9643, 0.9268)
AUCMatrix = data.frame(Weather_AllYears = Weather_AllYears_AUC,
                       Weather_SprayYears = Weather_SprayYears_AUC,
                       SprayWeather_SprayYears = SprayWeather_SprayYears_AUC)
rownames(AUCMatrix) = c("GLM", "GLM Elastic Net", "RandomForest", "Gradient Boosting", "AdaBoost")
```

ConfusionMatrices:

```
library(xtable)
Method = rep(c("GLM", "ElasticNet", "RandomForest", "GradientBoosting", "AdaBoost"), times=3)
Data = rep(c("AllYears, Weather", "SprayYears, Weather", "SprayYears, SprayWeather"), each=5)

#AUC
AllFitClassification = c(89.1, 10.5, .1, .3, 89.1, 10.6, .1, .1, 88.3, 4, .9, 6.7, 87.8, 6.2, 1.5, 4.6,
     88.4, 9.2, .8, 1.6)
SprayYrWeatherClassification = c(86.1, 12.5, .5, .9, 86.2, 12.7, .4, .7, 85, 3.2, 1.6, 10.2, 84.4, 5, 2.2,
      8.4, 84.6, 8.8, 2, 4.6)
SprayYrWeatherSprayClassification = c(86.1, 12.4, .5, 1, 86.1, 12.5, .5, .9, 85.1, 3.3, 1.5, 10.1, 84.5,
     5, 2.1, 8.4, 84.5, 9, 2.1, 4.4)
Confusions = matrix(c(AllFitClassification, SprayYrWeatherClassification,
     SprayYrWeatherSprayClassification), ncol=4, byrow=TRUE)
colnames(Confusions) = c("True Neg", "False Neg", "False Pos", "True Pos")
print(xtable(data.frame(Method, Data, Confusions)), include.rownames=FALSE)

#F1
AllFitClassification = c(89.1, 10.5, .1, .3, 89.1, 10.5, .1, .2, 88.3, 4, 1, 6.8, 87.9, 6.3, 1.4, 4.5,
     88.4, 9.1, .8, 1.7)
SprayYrWeatherClassification = c(86.1, 12.5, .5, .9, 86.5, 12.9, 0, .5, 85.2, 3.4, 1.4, 10, 85.3, 10.5,
     1.3, 2.9, 84.6, 8.8, 2, 4.7)
SprayYrWeatherSprayClassification = c(86.1, 12.4, .5, 1, 86.4, 12.8, .2, .6, 85.1, 3.3, 1.5, 10.1, 85.4,
     10.2, 1.2, 3.2, 84.5, 8.7, 2.1, 4.7)
Confusions = matrix(c(AllFitClassification, SprayYrWeatherClassification,
     SprayYrWeatherSprayClassification), ncol=4, byrow=TRUE)
F1 = sapply(1:nrow(Confusions), function(i) {
  sens = Confusions[i, 4] / (Confusions[i, 2] + Confusions[i, 4])
  ppv = Confusions[i, 4] / (Confusions[i, 3] + Confusions[i, 4])
```

```
  round(2 * sens * ppv / (ppv + sens), 3)
})
Confusions = data.frame(Confusions, F1)
colnames(Confusions) = c("True Neg", "False Neg", "False Pos", "True Pos", "F1")
print(xtable(data.frame(Method, Data, Confusions)), include.rownames=FALSE)

#BA
AllFitClassification = c(89.1, 10.5, .1, .3, 89.1, 10.5, .1, .2, 88.2, 3.9, 1, 6.8, 87.8, 6.6, 1.5, 4.1,
    88.4, 9.1, .8, 1.6)
SprayYrWeatherClassification = c(86.1, 12.5, .5, .9, 86, 12.5, .5, .9, 85, 3.2, 1.5, 10.2, 84.5, 5.2, 2.1,
    8.3, 84.6, 8.8, 2, 4.6)
SprayYrWeatherSprayClassification = c(86.1, 12.4, .5, 1, 86.1, 12.5, .5, .9, 84.9, 3.2, 1.7, 10.2, 84.5,
    5, 2.1, 8.4, 84.5, 9, 2.1, 4.4)
Confusions = matrix(c(AllFitClassification, SprayYrWeatherClassification,
    SprayYrWeatherSprayClassification), ncol=4, byrow=TRUE)
BA = sapply(1:nrow(Confusions), function(i)
  round(mean(c( (Confusions[i, 1] / (Confusions[i, 1] + Confusions[i, 3])),
  (Confusions[i, 4] / (Confusions[i, 2] + Confusions[i, 4])))), 3))
Confusions = data.frame(Confusions, BA)
colnames(Confusions) = c("True Neg", "False Neg", "False Pos", "True Pos", "BA")
print(xtable(data.frame(Method, Data, Confusions)), include.rownames=FALSE)
```

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3121–3124. IEEE, 2010.

[3] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[4] Wendy Kan. Kaggle west nile virus prediction: Spray effort. `https://www.kaggle.com/c/predict-west-nile-virus/forums/t/13724/spray-effort/74430#post74430`. Forum Post. Posted: 2015-04-28, Accessed: 2015-05-17.

[5] Wendy Kan. Kaggle west nile virus prediction: Spray effort. `https://www.kaggle.com/c/predict-west-nile-virus/forums/t/13724/spray-effort/75071#post75071`. Forum Post. Posted: 2015-05-02, Accessed: 2015-05-17.

[6] Max Kuhn. Building predictive models in r using the caret package. *Journal of Statistical Software*, 28(i05), 2008.

[7] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7, 2013.

[8] David Martin Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*, 2011.

[9] MO Ruiz, LF Chaves, GL Hamer, T Sun, WM Brown, ED Walker, L Haramis, TL Goldberg, and UD Kitron. Local impact of temperature and precipitation on west nile virus infection in culex species mosquitoes in northeast illinois, usa. *Parasites & Vectors*, 3(1):19–19, 2009.

[10] Michael R Sardelis, Michael J Turell, David J Dohm, and Monica L O'Guinn. Vector competence of selected north american culex and coquillettidia mosquitoes for west nile virus. *Emerging infectious diseases*, 7(6):1018, 2001.

[11] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning*, pages 80–91, 1999.

[12] Michael J Turell, Monica L OGuinn, David J Dohm, and James W Jones. Vector competence of north american mosquitoes (diptera: Culicidae) for west nile virus. *Journal of Medical Entomology*, 38(2):130–134, 2001.

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