**Web Material**

Title: Using Machine Learning Techniques and National Tuberculosis Surveillance Data to Predict Excess Growth in Genotyped Tuberculosis Clusters

Authors: Sandy P. Althomsons*, Kathryn Winglee*, Charles M. Heilig, Sarah Talarico, Benjamin Silk, Jonathan Wortham, Andrew N. Hill, Thomas R. Navin

# Table of Contents

# Web Appendix 1

**Analytic Overview**

1. Create two datasets: 2011–2015 data used for 5-fold cross-validation (training and testing) and 2016–2017 data for validation
2. Partition 2011–2015 data for 5-fold cross-validation: Data are partitioned into five non-overlapping subsets, each consisting of 20% of the data, using the R sample function with the seed set to 1234. Each data subset is used as a test dataset while the remaining four datasets (comprising 80% of the data) are used as the training set, for a total of 5 training/test set combinations.
3. For each of the five training sets and for each of the timeframe and quantification predictor combinations, run all machine learning methods (Table 1) on the training set and calculate performance metrics on the test set
4. Calculate the average performance metric from the five test sets
5. Select the model with the largest average Youden index
6. Perform hyperparameter tuning on selected model (results not shown)
7. Final model for future use is built on all data from the cross-validation using the model (machine learning method, timeframe and quantification) identified in step 5
8. Assess final model performance on validation set (2016–2017 data)

# Web Appendix 2

**Classification Methods**

For classification, our outcome was excess growth (positive class) or expected growth (negative class). Table 2 shows the number of clusters with excess growth in each. We then ran each training set through each of the models indicated in Table 1 with default settings.

Method performance was evaluated using the confusion matrix from the caret package[1], and averaged across the five runs:

| | | **Actual cluster status** | |
|---|---|---|---|
| | | **Excess growth** | **Expected growth** |
| **Predicted cluster status** | **Excess growth** | True positive (TP) | False positive (FP) |
| | **Expected growth** | False negative (FN) | True negative (TN) |

The metrics used by this study (calculated by the caret package) were:

$$\text{Specificity} = \frac{TN}{TN+FP}$$

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Youden index} = sensitivity + specificity - 1$$

$$\text{Positive predictive value} = \text{PPV} = \frac{TP}{TP+FP}$$

$$\text{Negative predictive value} = \text{NPV} = \frac{TN}{TN+FN}$$

We chose the model with the largest Youden index, after limiting to only those models with a mean sensitivity > 0.25. This cutoff was added so we would only select models that would predict at least a quarter of the clusters with excess growth. Web Table 1 contains the averaged results from the 5-fold cross-validation. Based on these results, we selected random forest with a 2Q timeframe and half quantification.

We then performed hyperparameter tuning by testing alternate values of the random forest hyperparameters (see details in section 4.1.2) using 5-fold cross-validation of the 2Q timeframe and half quantification. However, these changes resulted in minimal improvements in our metrics from the 5-fold cross-validation and performed poorly on the validation set, so the random forest default settings were used for our final model (data not shown).

All analyses were run in R version 4.0.2[2] and all code is available at https://github.com/CDCgov/Predicting_TB_cluster_growth.

# Web Appendix 3

**Validation and Model Interpretation**

Our final model was a random forest built using the randomForest package[3] on the 2Q timeframe with the half quantification (the model with the best performance from the cross-validation in Web Table 1) using all 332 observations used in the cross-validation. We then validated this final model performance by generating predictions on the validation data set (Table 3). In addition, we used this model to generate an ALE plot (Web Figure 1) and analyze the importance metrics (Figure 3) for each of the variables in the half predictor set to better understand which predictors are most informative and how those predictors relate to the probability of a cluster being predicted to have excess growth. These analyses show that the time between unexpected growth cases is the most important variable (Figure 3), and in general as the time increases, the accumulated average prediction of probability that excess growth will be predicted decreases (Web Figure 1). In other words, as the length of the time between unexpected growth cases increases, unexpected growth is less likely, which is in line with program experience. In contrast, the next three most important variables (baseline value, cases before baseline among prevalent clusters, and cases in quarter of unexpected growth), show the opposite trend, with an increasing variable value increasing the accumulated average probability that excess growth will be predicted, with the exception of a local minima at a baseline value of 0.25 and a local maximum at a value of 2 for cases before baseline among prevalent clusters. The proportion of clusters with excess growth in our training and testing dataset was lower (for baseline value) or higher (for cases before baseline) at these values than at other values for these variables, suggesting there might be something unique about these values. Note that for these variables the data tends to get sparser as the value increases (as represented by the ticks on the x-axis of Web Figure 1), so there is less support for the trend at the higher values. Likewise, among the categorical variables, if over half the cluster is U.S.-born, attributed to recent transmission, has cavitary results, was prevalent, was found through active case finding, reported non-injection drug use, or reported homelessness, then the accumulated average probability of a prediction that the cluster will have excess growth will increase. For the remaining categorical variables, the probability would decrease. Again, note that for some of these variables, such as pediatric cases or HIV positive test results, very few clusters have a value of 1, and so these effects must be interpreted with caution. However, these predictors also have very low importance.

# Web Appendix 4

## Tree-based ensembles

### Classification and Regression Trees (CART)

Classification and Regression Trees (CART), which are examples of decision trees, can be used to generate easily interpretable classification or regression models, where the data are recursively split at nodes based on a single variable[4]. The resulting tree is easy to understand, and the SAS JMP 9.0.1 implementation of decision trees was used in a previous version of this project[5]. Early analysis tested several CART functions, including recursive partitioning (rpart function) in the rpart package[6, 7], but these performed poorly compared to other models (results not shown), and so were not included in later iterations. However, CART as implemented in rpart is described here, as understanding CART is important to understanding the other tree functions, which build on CART.

CART generates classification or regression models that can be represented as binary trees. The tree is built in two steps. First, the single feature which best splits the data into two groups is identified and the data is separated. This process is then applied separately to each new subgroup, until the subgroups either reach a minimum size (5) or no improvement can be made. For classification, "best" is measured using Gini index:

$$\text{Gini index: } \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$$

$$\text{where } \hat{p}_{mk} = \frac{1}{N_m}\sum_{x_i \in R_m} I(y_i = k)$$

where $m$ is a node representing region $R_m$ with $N_m$ observations and I is the indicator function. In other words, $\hat{p}_{mk}$ is the proportion of class $k$ observations in node $m$, and the Gini index is a measure of the purity of the node (i.e. whether the node is mostly one class or another). Note that Gini index is the default for rpart, but other criteria could be used.

In the second step, cross-validation is used to trim the tree. Let $T_1, T_2, \ldots, T_k$ be the terminal nodes of a tree T. Then $|T|$ is the number of terminal nodes and the risk $R(T)$ of the tree $T$ is defined as $R(T) = \sum_{i=1}^{k} P(T_i)R(T_i)$. Then let $\alpha > 0$ (a complexity parameter) be a number which measures the cost of adding another variable to the model and define the increased risk as $R_\alpha(T)=R(T) +\alpha|T|$.The goal is to find $T_\alpha$, the smallest tree for which $R_\alpha(T)$ is minimized. 10-fold cross-validation is used to choose the $\alpha$ with the smallest risk, and the final tree is $T_\alpha$. In practice, the 1-SE rule is usually used, where any risk within one standard error of the achieved minimum is considered equivalent to the minimum and the simplest model that meets this criterion is chosen.

### Random Forest

Random forest builds on the ideas behind the classification and regression tree algorithm described above[8]. We used the random forest implementation in the randomForest[3] package. Instead of constructing a single tree, the algorithm builds a forest of N trees (default N in randomForest package used is 500 trees). For each tree in the forest, the root node is a bootstrap sample (random sample with replacement) of data of the same size as the original data (in contrast to CART, which looks at all data). Each tree is grown using a different bootstrap sample. At each node, K of the variables are selected at random and only these variables are searched through for the best split (in contrast to CART, which looks at all variables at each node). K must be less than the number of variables, and we used the randomForest default, which is the square root of the number of variables. The largest tree possible is grown and is not pruned. To classify a new observation **x**, **x** is run through each of the N trees, each of which gives a value for **x**. The forest then chooses the classification with the most votes (for classification) or the average of the N trees (for regression).

Given that random forest had the highest Youden index, therefore selected as the best model, we also tested changing several of the hyperparameters of the model built using the half quantification and 2Q timeframe. We tested changing N (number of trees; default is 500 but tested 400—1,000 in increments of 100), K (number of variables selected at each node; default with our data was 4 but tested 2—10 in increments of 1), and the minimum size of the terminal nodes (default is 1 but tested 1—5 in increments of 1).

**Gradient Boosting Machine (GBM)**

Boosting is a method of combining the performance of many weak classifiers into a strong classifier[9-11]. It works by sequentially applying an algorithm to reweighted versions of the training data and then taking a weighted majority vote of the classifiers. In other words, it applies a parameterized function (base learner) to current "pseudo"-residuals, which are the gradient of the loss function. We used default settings in the gbm (generalized boosted modeling) function of the gbm package[12], which implements Friedman's Gradient Boosting Machine. In this algorithm, $\boldsymbol{x}_i$ are the covariates, *N* is the number of observations, *T* is the number of iterations (trees), *K* is the depth of each tree, $\lambda$ is the shrinkage (learning) rate, $\rho$ is the gradient descent step size, *p* is the subsampling rate, and $\hat{f}(\boldsymbol{x}_i)$ is a regression function that minimizes the loss function $\Psi(y, f)$. This algorithm as implemented in gbm() is:

Start with $\hat{f}(\boldsymbol{x}) = \arg min_\rho \sum_{i=1}^{N} \Psi(y_i, \rho)$ . For *t* in 1,…,T:

1. Compute the negative gradient as the working response:

$$z_i = -\frac{\partial}{\partial f(\boldsymbol{x}_i)} \Psi(y_i, f(\boldsymbol{x}_i))\Big|_{f(x_i)=\hat{f}(x_i)}$$

2. Randomly select *p* x *N* cases
3. Using only the randomly selected observations, fit a regression tree with *K* terminal nodes
4. Using only the randomly selected observations, calculate the terminal node predictions:

$$\rho_k = \arg min_\rho \sum_{\boldsymbol{x}_i \in S_k} \Psi\big(y_i, \hat{f}(\boldsymbol{x}_i) + \rho\big)$$

   where $\rho_1, \ldots, \rho_K$ are the terminal node predictions and $S_k$ is the set of **x**s that define the terminal node *k*.
5. Update $\hat{f}(\boldsymbol{x})$ estimate:

$$\hat{f}(\boldsymbol{x}) \leftarrow \hat{f}(\boldsymbol{x}) + \lambda \rho_{k(x)}$$

   where an observation having features **x** would fall into the terminal node with index *k*(**x**)

**<u>Support Vector Machine</u>**

A support vector machine (SVM) is another technique that can be used for either regression or classification. Essentially, an SVM works by mapping the training data into a high-dimensional space and then trying to separate the data in that hyperplane. A kernel function can be used to warp that space to make that separation easier. We used LIBSVM in the e1071 R package with default settings, except that we created four separate models, each testing one of the four implemented kernels [13]:

- Gaussian radial basis function (RBF): $K\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0$
- Linear: $K\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Polynomial: $K\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = (\gamma \boldsymbol{x}_i^T \boldsymbol{x}_j + r)^d, \gamma > 0$
- Sigmoid: $K\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) = tanh(\gamma \boldsymbol{x}_i^T \boldsymbol{x}_j + r)$

In these equations, $\gamma, r,$ and *d* are kernel parameters, and $\boldsymbol{x}_i \epsilon R^n, i = 1, \ldots, l$ is the training set containing the features. $\gamma$ can be seen as the inverse of the radius of influence of samples selected by the model as support vectors, r is the independent term in the kernel function, and d is the degree of the polynomial kernel function.

At a high level, for classification, SVM tries to find a linear separating hyperplane with the maximal margin (maximal distance between datapoints of both classes) in this space. This is the maximum margin hyperplane. Note that the kernel function can be used to transform the original data into a linearly separable space. Support vectors are the data points closest to the hyperplane and most influence the position and orientation of the hyperplane. Mathematically, the key idea behind SVM is to construct a primal objective function, which can be solved using the dual formulation[14].

For classification, we used the C-support vector classification[15] with default settings implemented in LIBSVM, which solves the primal optimization problem:

$$\min_{w,b,\xi} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i$$

$$\text{subject to: } y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0, \;\; i = 1, \ldots, l$$

where $\boldsymbol{x}_i \epsilon R^n, i = 1, \ldots, l$ is the training vector, $\mathbf{y} \epsilon R^l$ is an indicator for excess or expected growth such that $y_i \epsilon \{1, -1\}$, $\phi(\boldsymbol{x}_i)$ maps $\boldsymbol{x}_i$ into a higher-dimensional space, and C > 0 is the regularization parameter. Larger values of C will encourage a smaller margin. This usually involves solving the dual problem:

$$\min_{\alpha} \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha}$$

$$\text{subject to: } \boldsymbol{y}^T\boldsymbol{\alpha} = 0, \;\; 0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, l$$

where $\mathbf{e} = [1,\ldots,1]^T$ is the vector of all ones, Q is an *l* by *l* positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $K(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ is the kernel function. LIBSVM uses a Sequential Minimal Optimization (SMO)-type decomposition[16] to solve this equation. Once this is solved, the decision function is:

$$\text{sign}(\boldsymbol{w}^T\phi(\boldsymbol{x}) + b) = \text{sign}(\sum_{i=1}^{l} y_i\alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b)$$

## Regularized Regression

Regularized regression is a technique designed to reduce model variance by introducing bias. It is used to constrain the coefficient estimates in a generalized linear model, shrinking them towards zero[17, 18]. In other words, it avoids overfitting data by penalizing learning more complex models, in particular by penalizing coefficients if they are too far from zero (imposes a penalty on their size), forcing them to be small. This allows inclusion of an arbitrarily large number of predictors while avoiding issues with model stability or overfitting. Thus, these methods are well-suited to problems with a potentially large number of predictors or predictors that might be collinear. We tested three kinds of regularization techniques: ridge regression, lasso regression, and elastic net regression. For all three, we used the glmnet function in the glmnet package with family argument set to binomial[18].

### Ridge Regression

Ridge regression sets an upper limit on the magnitudes of the coefficients, in particular the sum of the squared coefficients. Thus, in ridge regression, we are trying to minimize the quantity:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}{\beta_j}^2$$

where *N* is the number of observations (size of the training data), *p* is the number of coefficients, and λ ≥ 0 is the tuning parameter. Note that the first half of this is the equation for least squares regression, while the second half adds a penalty on the square of the magnitude of the coefficients. The tuning parameter λ is chosen by using a grid of λ values and computing the cross-validation error for each value of λ. The tuning parameter for which the cross-validation error is smallest is selected. Note that the intercept $\beta_0$ is not in the penalty term, and is set to the average *y* value ($\bar{y}$). Also note that if λ = 0, there will be no penalty and the result will be equal to a least squares regression. However, as λ increases, the shrinkage penalty grows and the coefficient estimates will decrease towards zero. The end result is that we minimize the sum of squared residuals but, due to the last term (the $L_2$ penalty), we also penalize the size of the parameter estimates, thus shrinking them towards zero. Variables with minor contributions will have small coefficients, but all variables are in the model.

**Lasso Regression**

Lasso (Least Absolute Shrinkage and Selection Operator) regression is similar to ridge regression in that it adds a penalty for non-zero coefficients. It also sets an upper limit on the magnitudes of the coefficients, in this case by limiting the sum of the absolute value of the coefficients. Thus, the lasso coefficients minimize the quantity:

$$\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Note that the main difference from ridge regression is the last term, which takes the absolute value of $\beta_j$ (the coefficients), rather than the square. Again, λ is the tuning parameter, and if zero will give the same results as a least squares regression. It is selected the same way as in ridge regression.

The last term (the $L_1$ penalty), has the effect of forcing some coefficient estimates to be zero, resulting in a sparse model that involves only a subset of the variables. Thus, one major difference between lasso and ridge is that lasso can force some coefficient estimates to be exactly zero while ridge will not generally make them exactly zero. As a result, lasso can also be used for feature selection. The end result is that only the most dominant variables are in the model.

**Elastic Net Regression**

Elastic net regression combines the penalties of ridge and lasso regression. It minimizes:

$$\frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\left[\frac{1}{2}(1-\alpha){\beta_j}^2 + \alpha|\beta_j|\right]$$

where $\alpha$ is the mixing parameter. Note that if $\alpha$ = 0, then you have the ridge penalty and if $\alpha$ = 1, you have the lasso penalty. The value of $\alpha$ was selected using the caret[1] package for $\alpha$-tuning, with 10-fold cross-validation, while λ is chosen as described above.

Note that if there are strong correlations among variables, lasso tends to choose one and ignore the rest while ridge tends to shrink the coefficients towards each other. In contrast, elastic net causes highly correlated features to be averaged while encouraging a sparse solution. The end result is that it shrinks some coefficients towards zero (like ridge regression) and sets some to exactly zero (like lasso regression).

## Model interpretation

### Variable importance

Random forest variable importance is used as a way to identify which predictors in a random forest model most contribute to a prediction[8]. For this study, variable importance was evaluated using the mean decrease in node impurity, as measured by the Gini index (described above in 4.1.1). In other words, it is the sum of all the Gini index decreases for a given predictor, normalized by the number of trees. The larger the decrease, the more influential the variable. We calculated this using the importance function built into the randomForest package[8].

### Accumulated local effects (ALE) plots

ALE plots are a method of assessing how a predictor affects the prediction of the model on average, computed over the conditional distribution of that predictor[19, 20]. ALE plots were chosen over other options to visualize the main and interaction effects of predictors because they are fast to compute and are unbiased when predictors are correlated. They do this by calculating the local effect of the predictor on the model, then averaging this local effect across all values of the other predictors before accumulating (integrating) the averaged local effect. Each local effect is calculated by dividing the predictor space into intervals, and for each of the data instances in that interval, a prediction is made, replacing the predictor with the upper and lower limit of the interval. The difference between these two limits is calculated, accumulated, and centered, resulting in an ALE curve. These plots can be interpreted as accumulating the local effects of a predictor to visualize the underlying global effect.

ALE values are estimated by first estimating the uncentered effect:

$$\hat{g}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i:x_{i,j}\in N_j(k)\}} \left[f\left(z_{k,j}, \mathbf{x}_{i,\setminus j}\right) - f\left(z_{k-1,j}, \mathbf{x}_{i,\setminus j}\right)\right]$$

where *f*(x) is the fitted model that predicts the probability that *Y* falls into a particular class (for classification) as a function of the response variable **X** (consisting of *d* predictors). *K* is the number of intervals, with *k* indicating the interval index, while *j* is the predictor index (ranges from 1 to *d*). For each $j \in \{1,2,\dots,d\}$, $N_j(k) = \left(z_{k-1,j}, z_{k,j}\right]: k = \{1,2,\dots,K\}$ is a partition of the sample $\mathrm{x}_{i,j}: i = 1,2,\dots,n$ into *K* intervals. $z_{k,j}$ is the $\frac{k}{K}$ quantile of $\mathrm{x}_{i,j}$, with $z_{0,j}$ the smallest observation and $z_{K,j}$the largest observation. $n_j(k)$ is the number of observations in the *k*th interval $N_j(k)$. $k_j(x)$is the interval into which *x* falls. $\mathbf{x}_{i,\setminus j}$ is the *i*th observation of the subsets of predictors $\mathbf{X}_{\setminus J}$where $j = 1,2,\dots,d; j \neq J$.

Essentially, this method calculates differences in predictions, where the predictor is replaced with grid values *z*. The difference in prediction for a single instance in that interval is the effect, and the effects are summed up for all data instances in that interval (the neighborhood $N_j(k)$) in the second summation. This sum is divided by the number of instances in that interval to get the average difference in predictions for that interval (the local average). The left sum then accumulates these average effects across all intervals. Intervals are selected using the quantiles of the distribution of the predictor, to ensure the same number of data instances in each interval. Finally, special consideration is needed for

categorical features. Since predictor values need to have an order, for categorical features, the order is based on similarity to other features. The distance is calculated using the Kolmogrov-Smirnov distance for numerical features or relative frequency tables for categorical features, then multi-dimensional scaling reduces the distance matrix to one dimension to get a similarity-based order of the categories.

Once this ALE main effect estimator is calculated, it is then centered so the mean effect is zero:

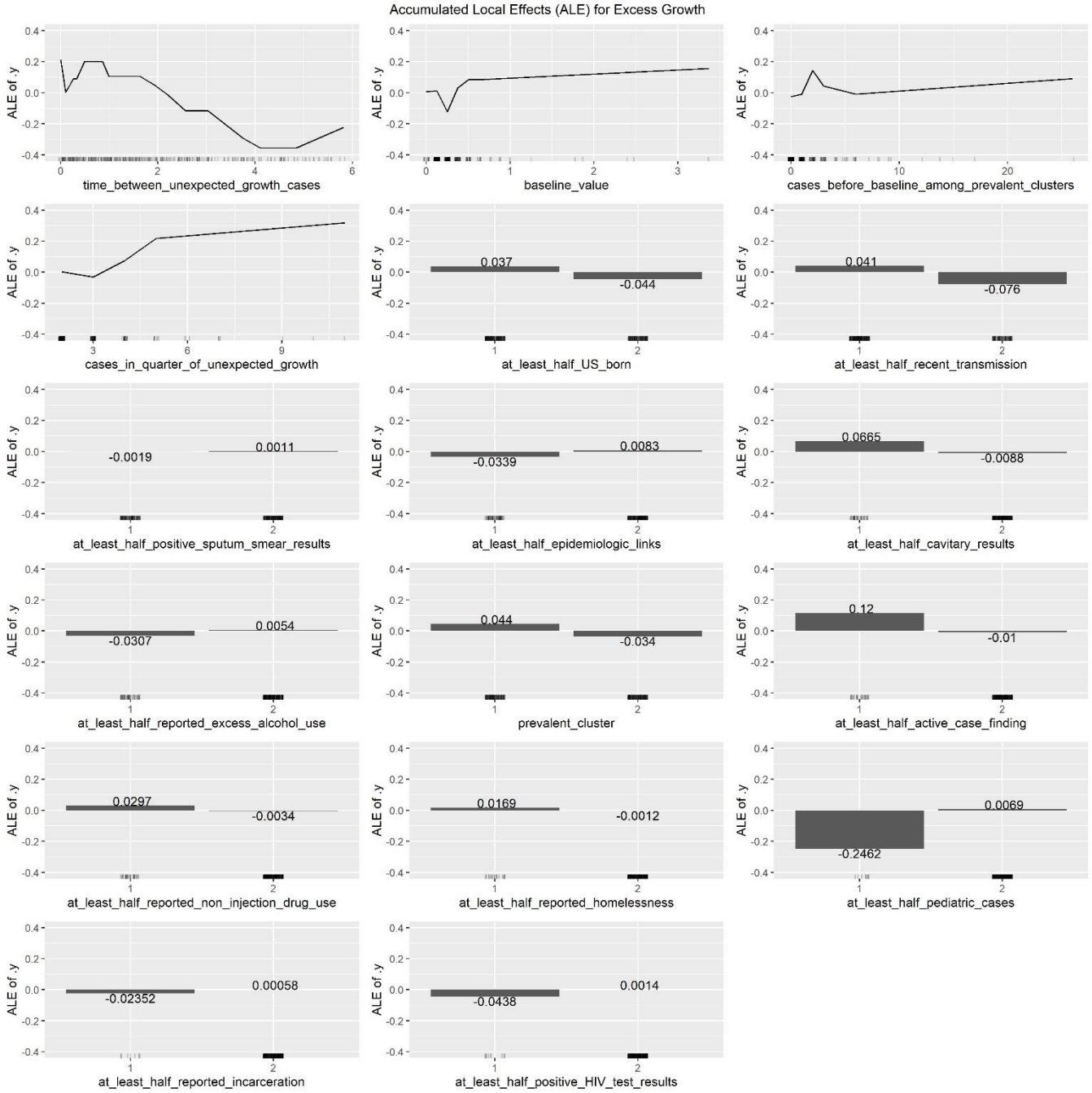
$$\hat{f}_{j,ALE}(x) = \hat{g}_{j,ALE}(x) - \frac{1}{n}\sum_{i=1}^{n} \hat{g}_{j,ALE}(x_{i,j})$$

Thus, an ALE value can be interpreted as the main effect of the feature at a given value compared to the average prediction of the data. In other words, the ALE plot gives the relative effect of changing the predictor; because ALE plots are centered at zero, the value of each point on the ALE curve is the difference from the mean prediction. We generated ALE plots using the FeatureEffects function in the iml package[21].

# References


1. Khun M. caret: Classification and Regression Training. R package 6.0-94. 2019. https://CRAN.R-project.org/package=caret. Accessed August 14, 2020.
2. R Core Team. R: A language and environment for statistical computing, version 4.0.2. Vienna, Austria: R Foundation for Statistical Computing; 2019.
3. Liaw A, Wiener M. Classification and Regression by randomForest. R News 2002; 2: 18-22.
4. Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and Regression Trees. Boca Raton, FL: Chapman & Hall/CRC, 1984.
5. Althomsons SP, Kammerer JS, Shang N, Navin TR. Using routinely reported tuberculosis genotyping and surveillance data to predict tuberculosis outbreaks. PLoS One 2012; 7(11): e48754.
6. Therneau T, Atkinson B. rpart: Recursive Partitioning and Regression Trees. R package version 4.1-15. 2019. https://CRAN.R-project.org/package=rpart. Accessed August 14, 2020.
7. Therneau TM, Atkinson EJ. An Introduction to Recursive Partitioning Using the RPART Routines: Mayo Clinic, 1997.
8. Breiman L. Random forests. Machine Learning 2001; 45: 5-32.
9. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting. Annals of Statistics 2000; 28(2): 337-74.
10. Friedman JH. Greedy function approximation: A gradient boosting machine. Annals of Statistics 2001; 29(5): 1189-232.
11. Friedman JH. Stochastic gradient boosting. Computational Statistics and Data Analysis 2002; 38(4): 367-78.
12. Greenwell B, Boehmke B, Cunningham J, GBM Developers. gbm: Generalized Boosted Regression Models. R package version 2.1.5. 2019. https://CRAN.R-project.org/package=gbm. Accessed August 14, 2020.
13. Chang C-C, Lin C-J. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2011; 2: 1-27.
14. Smola AJ, Scholkopf B. A tutorial on support vector regression. Statistics and Computing 2004; 14: 199-222.
15. Cortes C, Vapnik V. Support-Vector Networks. Machine Learning 1995; 20: 273-97.
16. Fan R-E, Chen P-H, Lin C-J. Working Set Selection Using Second Order Information for Training Support Vector Machines. Journal of Machine Learning Research 2005; 6: 1889-918.
17. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York, NY: Springer, 2009.
18. Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. J Stat Softw 2010; 33(1): 1-22.
19. Apley DWZ, Jingyu. Visualizing the effects of predictor variables in black box supervised learning models. Journal of the Royal Statistical Society Series B (Statistical Methodology) 2020; 82: 1059–86.
20. Molnar C. Interpretable machine learning: A Guide for Making Black Box Models Explainable: https://christophm.github.io/interpretable-ml-book/, 2019.
21. Molnar CB, Bernd; Casalicchio, Giuseppe. iml: An R package for Interpretable Machine Learning. Journal of Open Source Software 2018; 3: 786.

Figure and Table




## Web Figure 1. Accumulated local effects (ALE) plots for all predictors in final model.

The final model was built using random forest on the 2Q timeframe with the half quantification. We then calculated ALE plots for each of the predictors from this analysis. Predictors are shown from high to low importance (same order as Figure 3). In each plot, the y-axis is the ALE main effect of that predictor (the predictor shown is indicated below the plot) on the probability of predicting excess growth (in other words, the y-axis indicates how the prediction of excess growth changes locally when the predictor shown in the plot is varied). The x-axis is the value of the predictor, and the black ticks indicate the distribution of the predictor's values. Little or no tick marks mean that there is little data in that region

and so that region should be interpreted cautiously. For the categorical variables (represented as bars instead of lines), on the x-axis, 1 indicates an observation has the characteristic while 2 indicates that it does not. To better visualize the smaller effects for the categorical variables, numbers above/below bars indicate the actual ALE value. Little or no tick marks mean that there is little data in that region and so that region should be interpreted cautiously.

| timeframe | quantification | method | Youden index | accuracy | sensitivity | PPV | specificity | NPV |
|---|---|---|---|---|---|---|---|---|
| 2Q | half | random forest | 0.165(0.047) | 0.608(0.033) | 0.398(0.106) | 0.563(0.061) | 0.768(0.085) | 0.637(0.056) |
| 9Q | percent | SVM with linear kernel | 0.149(0.072) | 0.593(0.037) | 0.386(0.126) | 0.555(0.120) | 0.763(0.110) | 0.630(0.089) |
| 9Q | any | random forest | 0.142(0.133) | 0.593(0.079) | 0.402(0.092) | 0.526(0.109) | 0.740(0.062) | 0.628(0.103) |
| 9Q | percent | elastic net | 0.137(0.136) | 0.594(0.068) | 0.331(0.156) | 0.567(0.178) | 0.806(0.115) | 0.624(0.099) |
| 9Q | any | elastic net | 0.134(0.154) | 0.600(0.072) | 0.294(0.198) | 0.645(0.232) | 0.840(0.110) | 0.621(0.104) |
| 4Q | mix | elastic net | 0.109(0.063) | 0.593(0.055) | 0.274(0.136) | 0.584(0.109) | 0.835(0.129) | 0.612(0.072) |
| 1Q | half | random forest | 0.091(0.136) | 0.572(0.062) | 0.367(0.140) | 0.480(0.082) | 0.724(0.025) | 0.614(0.071) |
| 2Q | any | random forest | 0.085(0.102) | 0.569(0.058) | 0.356(0.103) | 0.486(0.063) | 0.729(0.060) | 0.609(0.072) |
| 4Q | mix | SVM with linear kernel | 0.085(0.101) | 0.560(0.056) | 0.373(0.127) | 0.496(0.126) | 0.712(0.140) | 0.609(0.056) |
| 9Q | mix | SVM with linear kernel | 0.080(0.088) | 0.563(0.065) | 0.397(0.055) | 0.485(0.057) | 0.683(0.111) | 0.604(0.099) |
| 9Q | any | SVM with linear kernel | 0.071(0.139) | 0.554(0.068) | 0.363(0.130) | 0.477(0.122) | 0.708(0.092) | 0.604(0.103) |
| 2Q | mix | SVM with linear kernel | 0.068(0.113) | 0.557(0.059) | 0.364(0.120) | 0.467(0.076) | 0.704(0.072) | 0.604(0.072) |
| 9Q | half | random forest | 0.062(0.104) | 0.557(0.067) | 0.343(0.078) | 0.467(0.090) | 0.719(0.047) | 0.600(0.095) |
| 9Q | percent | random forest | 0.059(0.070) | 0.560(0.064) | 0.337(0.077) | 0.474(0.059) | 0.721(0.091) | 0.598(0.088) |
| 1Q | any | random forest | 0.058(0.096) | 0.557(0.048) | 0.339(0.098) | 0.460(0.061) | 0.719(0.020) | 0.600(0.055) |
| 4Q | half | random forest | 0.058(0.114) | 0.552(0.057) | 0.347(0.095) | 0.473(0.126) | 0.710(0.116) | 0.598(0.051) |
| 9Q | percent | SVM with sigmoid kernel | 0.056(0.092) | 0.560(0.044) | 0.289(0.050) | 0.489(0.144) | 0.767(0.089) | 0.596(0.066) |
| 9Q | mix | random forest | 0.055(0.051) | 0.560(0.055) | 0.340(0.071) | 0.469(0.058) | 0.714(0.081) | 0.597(0.083) |
| 9Q | mix | elastic net | 0.043(0.095) | 0.551(0.078) | 0.308(0.123) | 0.493(0.120) | 0.735(0.159) | 0.590(0.093) |
| 4Q | any | random forest | 0.038(0.107) | 0.548(0.052) | 0.304(0.079) | 0.460(0.122) | 0.733(0.083) | 0.591(0.064) |
| 4Q | percent | random forest | 0.037(0.138) | 0.542(0.071) | 0.319(0.123) | 0.461(0.127) | 0.719(0.115) | 0.591(0.085) |
| 4Q | mix | random forest | 0.031(0.096) | 0.536(0.054) | 0.321(0.148) | 0.443(0.078) | 0.710(0.125) | 0.590(0.079) |
| 4Q | percent | SVM with linear kernel | 0.027(0.107) | 0.551(0.041) | 0.258(0.107) | 0.444(0.142) | 0.769(0.061) | 0.588(0.053) |
| 4Q | percent | GBM | 0.025(0.069) | 0.479(0.024) | 0.765(0.118) | 0.429(0.056) | 0.260(0.087) | 0.609(0.096) |
| 4Q | half | GBM | 0.020(0.125) | 0.479(0.051) | 0.753(0.123) | 0.429(0.042) | 0.267(0.137) | 0.587(0.177) |
| 2Q | percent | random forest | 0.017(0.085) | 0.536(0.043) | 0.300(0.106) | 0.438(0.069) | 0.717(0.094) | 0.584(0.058) |
| 4Q | any | GBM | 0.016(0.130) | 0.476(0.058) | 0.740(0.081) | 0.428(0.058) | 0.276(0.088) | 0.584(0.142) |
| 4Q | mix | GBM | 0.012(0.085) | 0.476(0.028) | 0.732(0.083) | 0.426(0.043) | 0.280(0.098) | 0.577(0.125) |
| 1Q | mix | SVM with linear kernel | 0.005(0.084) | 0.527(0.043) | 0.339(0.089) | 0.421(0.045) | 0.667(0.048) | 0.581(0.052) |
| 1Q | percent | SVM with linear kernel | 0.002(0.176) | 0.536(0.090) | 0.264(0.158) | 0.399(0.138) | 0.738(0.055) | 0.582(0.084) |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2Q | percent | GBM | -0.001(0.042) | 0.464(0.033) | 0.723(0.059) | 0.422(0.039) | 0.276(0.068) | 0.576(0.054) |
| 9Q | mix | GBM | -0.001(0.129) | 0.476(0.081) | 0.694(0.116) | 0.423(0.103) | 0.305(0.109) | 0.580(0.082) |
| 2Q | mix | GBM | -0.013(0.091) | 0.455(0.047) | 0.736(0.100) | 0.416(0.052) | 0.250(0.038) | 0.577(0.092) |
| 1Q | percent | random forest | -0.014(0.103) | 0.518(0.058) | 0.325(0.084) | 0.412(0.062) | 0.662(0.082) | 0.573(0.058) |
| 1Q | mix | GBM | -0.015(0.082) | 0.449(0.037) | 0.771(0.106) | 0.416(0.035) | 0.213(0.057) | 0.575(0.096) |
| 2Q | mix | random forest | -0.017(0.104) | 0.518(0.052) | 0.290(0.073) | 0.422(0.107) | 0.692(0.117) | 0.570(0.049) |
| 1Q | percent | GBM | -0.017(0.082) | 0.455(0.040) | 0.733(0.112) | 0.415(0.047) | 0.250(0.056) | 0.575(0.063) |
| 9Q | percent | GBM | -0.044(0.080) | 0.455(0.056) | 0.679(0.106) | 0.407(0.096) | 0.277(0.071) | 0.548(0.049) |
| 1Q | any | GBM | -0.051(0.126) | 0.431(0.064) | 0.738(0.114) | 0.405(0.048) | 0.211(0.080) | 0.529(0.147) |
| 9Q | any | GBM | -0.055(0.066) | 0.449(0.059) | 0.655(0.095) | 0.403(0.093) | 0.290(0.083) | 0.536(0.042) |
| 9Q | half | GBM | -0.058(0.077) | 0.440(0.047) | 0.702(0.073) | 0.403(0.076) | 0.241(0.073) | 0.522(0.109) |
| 2Q | any | GBM | -0.077(0.130) | 0.431(0.073) | 0.673(0.105) | 0.396(0.076) | 0.250(0.079) | 0.513(0.100) |
| 1Q | mix | random forest | -0.103(0.078) | 0.476(0.040) | 0.261(0.114) | 0.330(0.057) | 0.636(0.060) | 0.542(0.051) |
| 1Q | half | GBM | -0.107(0.079) | 0.413(0.037) | 0.658(0.077) | 0.385(0.038) | 0.235(0.035) | 0.488(0.066) |
| 2Q | half | GBM | -0.120(0.087) | 0.401(0.055) | 0.697(0.074) | 0.384(0.062) | 0.183(0.036) | 0.455(0.060) |

## Web Table 1. Cross-validation test results for all models with sensitivity > 0.25.

Timeframe indicates which cases were included in the predictors: 1Q includes those cases in the flagged unexpected growth quarter, 2Q the flagged unexpected growth quarter and one preceding quarter, 4Q the flagged unexpected growth quarter and three preceding quarters, and 9Q the flagged unexpected growth quarter and eight preceding quarters. Quantification indicates how the predictors were classified: ‘any’ means at least one case in the cluster was positive for the characteristic, ‘half’ means at least half the cases in the cluster were positive for the characteristic, ‘percent’ indicates the percentage of cases in the cluster positive for the characteristic, and ‘mix’ is a combination of the other three. The metrics are then calculated from a 5-fold cross-validation (CV) of 332 observations, and averaged across the 5 CV runs, with the mean (standard deviation) indicated here. PPV= positive predictive value, NPV = negative predictive value, SVM = support vector machine, GBM = gradient boosting machine. Results were subsetted to models with a mean sensitivity > 0.25, and then sorted from high to low Youden index.

| Predictor (feature) | Description |
|---|---|
| Time between unexpected growth cases | The number of months between the first case reported during time period of interest (i.e., 1Q, 2Q, 4Q, or 9Q) and the last case reported during quarter of unexpected growth (QUG). |
| Baseline value | The average number of cases reported during the 8 quarters prior to the detection of unexpected growth. |
| Number of cases before baseline | The number of cases reported in two-year period prior to baseline period (two years prior to QUG) for clusters defined as prevalent, or zero for clusters defined as incident. |
| Cases in quarter of unexpected growth | The number of cases reported during QUG. |
| At least half US-born | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported being born within the United States (including U.S. territories) or born to at least one U.S. citizen parent. |
| At least half recent transmission | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported as attributable to recent transmission. |
| At least half positive sputum smear results | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported with positive sputum smear results. |
| At least half epidemiologic links | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported with an epidemiologic link to another TB case. |
| At least half cavitary results | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported with cavitary imaging results. |
| At least half reported excess alcohol use | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported excess alcohol use within past 12 months of TB diagnosis. |
| Incident cluster | Cluster of cases with same genotype in which the first case or beginning of possible outbreak could be identified, i.e., first case preceded by two years of no cases reported with same genotype in same jurisdiction. |
| At least half active case finding | Binary classification of cluster based on whether ≥50% of all cases during time period of interest (1) reported with reason for TB evaluation either because of targeted testing or because of a contact investigation, or (2) case reported with epidemiologic link to another case. |
| At least half reported non-injection drug use | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported non-injecting drug use within past 12 months of TB diagnosis. |
| At least half reported homelessness | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported experiencing homeless within past 12 months of TB diagnosis. |
| At least half pediatric cases | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported being <15 years of age at TB diagnosis. |
| At least half reported incarceration | Binary classification of cluster based on whether ≥50% of all cases during time period of interest reported being resident of correctional facility at time of TB diagnosis. |
| At least half positive HIV test results | Binary classification of cluster based on whether ≥50% of all cases during time period of interest had documented positive HIV test result. |

## Web Table 2. Summary of predictors used in final model.

Each predictor, or feature, that was included in the final model (selected as the one with the highest Youden index) is listed and described.