RoboStalin Methodology

0.6 Data Analysis

First we identified the most effective classifier algorithm for the dataset using cross validation. Cross validation allows us to estimate the best model before running the test data through it by "holding out" a subset of the training observations and then using them as a stand-in to test. The Cross-Validation method used here is k-Fold validation, which splits the observations into k groups, with k-1 groups used to train the model and the last group used as the test set. This is then repeated k times, with the cross validation estimate computed based on the average of the k test groups. For the purposes of this analysis, k was set to 5.

The classifiers that were tested were:

1. Naive Bayes- a probabilistic classifier based on the Bayes theorem. It calculates the probability that there is backsliding, given the values of the independent variables, assuming that the independent variables are also independent from one another
2. K Nearest Neighbors- a classifier that identifies the number of 'K' points from the training data that are closest to the observation of interest, and then determines its category based on the majority category of the nearest points.
3. Decision Trees- A classifier which sorts observations by splitting them based upon specific decision criteria. It then predicts that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs
4. Random Forests- A classifier which builds several different decision trees by pulling several training sets from the training data along with a random number of predictors. It then creates its decision criteria by averaging across the predictions from each tree
5. Support Vector Machine- A classifier which sorts objects into categories by creating “decision boundaries” distinguishing between classes.

In order to evaluate the performance of each algorithm, the full dataset was split into two subsets: a training set and a testing set. The algorithm was first fit on the training set, and then based on the information it gained from the training data it predicted the categories of the test set. The model was then judged on its ability to correctly the predict the dependent variables of the test set. The segregation of training data from testing prevents overfitting so model performance is accurately scored.

The Random Forest algorithm outperformed all others across all models in each dataset. A Random Forest model is fundamentally based on Decision Trees. Decision trees consist of a series of splits on the original dataset, known as a tree. Each split, which can be thought of as a branch, is made on a “decision node.” This decision node denotes a criteria that the data is being split on, for example countries with a GDP per capita greater than 25,000. In that countries with a GDP greater than 25,000 would go on one branch, while countries with GDP less than 25,000 would go on another branch. These groups would then be split on another criteria. The farthest branches of these trees are referred to as “terminal nodes” or leaves. In this way a decision tree is actually like an upside down tree, with the leaves on the bottom. The model utilizes a “top down” and “greedy” approach in deciding what features to split on. Put simply, this means that starting from the top of the tree, the algorithm makes each split based on what best minimizes classification errors at that specific step (ie grouping as many 1s together and 0s together as possible while minimizing the number of members of the other class in the group).

While Decision Trees have the advantage of being both relatively easy to interpret and a closer approximation of human decision making, they lack the predictive accuracy of other models, are sensitive to small changes in the data, and tend to overfit, especially as the tree gets deeper and decisions are made on smaller and smaller subsamples of the data. The Random Forest model improves on decision trees by building many trees, each created from a sub sample of the data, then merging them together and taking the most common predictions across each terminal node for each tree. Each tree is based on a random subset of the independent variables. The added variation caused by training on different samples of the data and utilizing various subset of the variables allows the model to create a more accurate and stable prediction upon taking the average of all the trees. The use of random subsets which are later combined also helps prevent overfitting. Additionally, the algorithm was tuned on several parameters in order to maximize its predictive power. These parameters are the maximum depth of each tree, the number of trees to build, and maximum number of features to consider for each tree. The ideal parameters differed across countries and y variables.