Forecasting GNI Per Capita

CS 6140 - Machine Learning
Spring 2017

Authors

Group #11

Introduction

Forecasting a given country's gross national income (GNI) per capita for the next year based on previous year's data and economic indicators is a useful tool for agencies such as the IMF, World Health Organization, and agencies which invest in private efforts in developing countries. Gross national income is equal to the gross domestic product in addition to income from foreign investments minus the income earned in the given country by nonresidents. The IMF collects data to compute various economic indicators for a subset of countries every year, ranging from urban population and percentage of households with running water to livestock production and merchandize exports. Although the data on countries is sparse due to the expense of collecting such data, there are a sufficient number of predictors (45) shared between all countries for a forty year period from 1971 to 2010 to train machine learning models to effectively forecast the next year's GNI as a function of the available data as of the current year.

Methods

Data

The original data was formatted as tuples consisting of country, year, economic indicator name and the value for the given economic indicator. The data was preprocessed by grouping on the composite key resulting from the country and year. The country name, year and indicators in the grouped result were placed along the horizontal axis as columns and their respective values were filled in from the group-by result. Three cleaned versions of the data were created. The first version included all predictors and countries available at this point with nulls as existed in the original data. The second and third versions required a user-defined minimum number of examples for each predictor. In the second version, all missing values were imputed using k nearest neighbors with the neighbor count equal to the required minimum number of observations. In the third version, predictors without sufficient year country pairs were dropped. Remaining rows with null values were dropped as well. All cleaned outputs contained a column with the next year's GNI. A final constraint was defined on all datasets that required the subset of countries selected to have a configurable minimum consecutive number of years worth of data in common. This interval was selected automatically to maximize the number of countries in the final dataset. All years outside of the selected interval were dropped in all datasets. This procedure resulted in the equal representation of each country. The shared time period enabled the modeling of latent variables for multiple countries at a shared point in time. The final dataset is partitioned into training and validation sets given a configurable percentage of data appearing in the validation set. The data is split such that the training and validation sets are contiguous sets of year with the validation set beginning on the year after the final year of the training set. Both sets contain the same countries.

1. Recurrent Neural Networks (RNN)

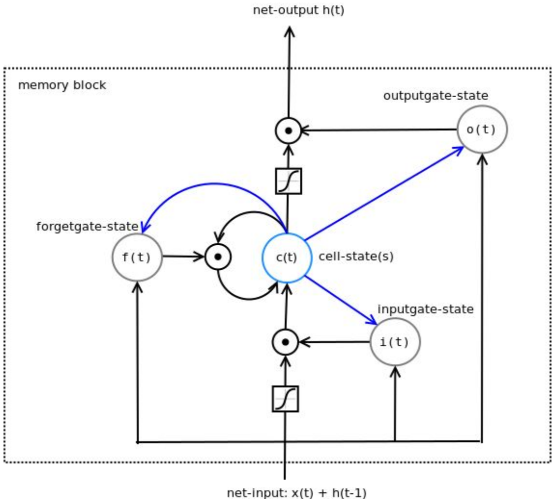
1.a Summary

A recurrent neural network is similar to a feed-forward neural network in that each neuron typically uses a sigmoid activation function and the input to each neuron is the dot product of the previous nodes' outputs and the weights associated with the connection plus all of the biases for the incoming connections. Feed-forward neural networks are typically well-suited to problems in which the current state is entirely defined by the input. Examples of such problems include classification and regression problems in which all of the observations are independent and identically distributed. Traditional feed-forward neural networks are not well-suited to modeling dependent observations because previously observed values are not retained by the model after the weights and biases have been finalized after training. Problems such as time-series forecasting violate the assumptions of feed-forward neural networks and require a more complex architecture to model effectively.

The theoretical recurrent neural networks relax the directed acyclic graph constraint on the feed-forward neural network connections, allowing directed loops among the connections in the network. These loops enable the network to retain previously observed values over a variable number of future steps. The presence of loops prevents the network from being evaluated entirely for a given input. The network is therefore evaluated for a finite number of steps before the values in the output layer are finalized.

Feed-forward neural networks are trained by defining a function which models the difference between the generated and expected output of the network. This function is known as the cost function and is defined such that it increases as the difference between the generated and correct values increases. The only additional constraint on the cost function is that it must be valid if averaged across multiple training batches. After propagating a training example through the network, the partial derivative of the cost function with the respect to each of the input weights and biases of the connections closest to the input layer is obtained. The computed gradients for each input x are used to update the weights and biases by a configurable learning rate in the direction in which the cost function is decreasing the most rapidly. This process is repeated until the cost function is minimized over the training set or another terminating condition is reached.

The theoretical recurrent neural network is impractical because training via backpropagation as feed-forward neural networks are trained results in the vanishing gradient problem if the partial derivative of loops are evaluated until the gradient converges. The gradients of the weights and biases of connections leading into the loops vanish (approach 0). A typical workaround involves unrolling all loops in the network for a configurable number of steps. The training inputs and outputs are paired. As many consecutive pairs are selected as the number of times the network has been unrolled. The network is then trained using a modified backpropagation procedure in which the final delta of the weights and biases is averaged across all computed gradients for the input steps. This procedure is known as backpropagation through time.



A flexible and widely used recurrent neural network architecture is one composed of long short term memory cells. Long short term memory cells were invented by Hochreiter and Schmidhuber (4) to address the vanishing gradient problem when training recurrent neural networks via backpropagation. A long short term memory cell is composed of a forget gate, previous input gate, previous output gate and a function which aggregates the current input with these components to produce the output for the current step. The activation functions for each of these nodes is configurable. The logistic function is used in this analysis. The ⊙ operator denotes the Hadamard product of the input elements. The directed lines connecting the elements are each associated with a weight and bias that are trained as described above. The model used in the analysis presented here is composed of a single layer of LSTM blocks. The initial weights and biases are randomly initialized to reduce bias in the model.

1.b Preprocessing

Although neural networks can in theory handle non-normalized data, it takes the network a long time to converge to an optimal set of weights and biases, especially with a large number of steps or deep networks. To enable the optimization to converge to an optimum more efficiently, the predictors and outputs are normalized using the maximum absolute value for each predictor. A maximum absolute value normalizer is preferable to a min max normalizer in that future data falling outside of the fitted range will not change sign when using a maximum absolute value normalizer. As a heuristic, the maximum absolute scaler is fit to the training data and the maximum per feature is doubled. It is possible that future inputs are larger than the max range defined by the training set, although cross validation on the training set did not indicate any issues with this dataset.

1.c Data Structure

The data is batched by country. One batch of size equal to the number of distinct years is trained at once. Each batch entry contains a three dimensional input and a one dimensional output column. The shape of the input data has observations along the z axis, predictor values along the x axis and the past n observations along the y axis. The number of steps n is defined to be ten. Although larger number of steps can in theory capture more detail, gradient descent takes longer to converge with a larger number of steps due to the vanishing gradient problem. Ten was chosen via experimentation because it is small enough to enable efficient training while reducing the likelihood of lost relevant information. It is unlikely that events more than ten years ago have a significant impact on the next year's GNI per capita. The segment of the data which was not labeled for validation was divided into a training and test set. The first half was used for training and the second used for testing. Cross validation was performed on folds of the training set. Model selection was performed on the test set.

1.d Results

A grid search to select the model hyperparameters such as network dimensions and number of epochs via cross-validation was attempted but impractical due to available compute resources. These hyperparameters were chosen via experimentation. The dropout rate regularization parameter was selected via five-fold cross validation on the training data set.

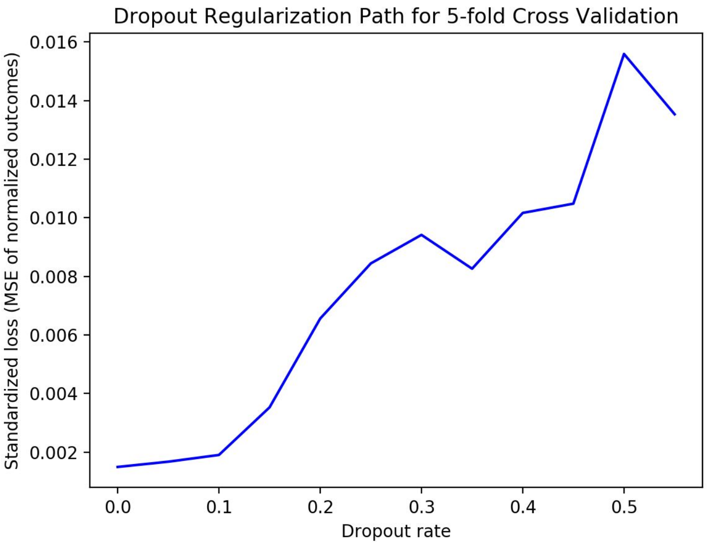


Fig. 1 - Dropout regularization path on 5-fold time-series cross validation. A dropout rate of 0 was selected.

The number of epochs which yields the best performance during experimentation was two-hundred. The number of layers which yields the best performance is one. Successive layers result in gradient descent taking longer to reach a globally optimal solution as a result of the vanishing gradient problem. The vanishing radient problem is particularly exacerbated by the unrolled model for each of the ten steps used. A hidden height of 400 was chosen because it reduced overfitting without a large dropout regularization parameter (0 was selected) while offering good performance on the test section of the training set. Two cost functions were evaluated: mean percent difference and mean squared error. The mean percent difference cost function skewed weights lower than expected because it penalized small differences in countries with low GNI per capita at the same rate as larger differences present in countries with large GNI per capita. Mean squared error was chosen as an alternative and the resulting model yields comparable absolute performance across all countries without the bias present as a result of the mean percentage difference cost function.

Dataset	Train RMSE	Test RMSE	Test Std.	Average Test Std. Per country
Training	255	709	11,267	1,498
Validation	214	705	14,297	1,643

The model outperforms the baseline standard deviation in the dataset by a large margin and outperforms the average standard deviation per country, indicating that it is useful in forecasting upcoming GNI given an arbitrary future time and a country in the original dataset. The data does indicate significant overfitting occurred. Although the dropout regularization rate of zero was selected on the training set via cross validation, the path was not very smooth. The discontinuities in the path indicate that a larger epoch size might have enabled a non-zero dropout rate to achieve the same perform as the network configured with a dropout rate of zero, reducing the performance drop from the training to test sets. The training RMSE of the training and validation sets are different because the model used on the validation set is trained using all data not labeled as validation data. The model evaluated prior to testing on the validation set is trained on a subset of the training data and evaluated on the remainder as described previously.

2. Tree-based methods

2.a Summary

2.a.a Introduction

Tree-based methods for regression and classification involves segmenting the predictor space into a number of regions. In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs. Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *Decision Tree* methods. Tree-based methods are simple and useful for interpretation. However, they typically are not competitive with the best supervised learning approaches, hence we have also used *Random Forests,* to improve the accuracy of the model.

2.a.b Basics of Decision Trees and Random Forests

Decision trees can be applied to both classification and regression problems, we have used it for our regression problem. It consists of a series of splitting rules, starting at the top of the tree. At the end, we are left with n splits such that the predictor space is divided into R1, R2, R3, … ,Rn. These regions are distinct and non-overlapping. In keeping with the tree analogy, the regions are known as *terminal nodes* or *leaves* of the tree. The points along the tree where the predictor space is split are referred to as *internal nodes*. For every observation that falls into the region Rj , we make the same prediction, which is simply the mean of the response values for the training observations in Rj. In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model. The goal is to find boxes R1, ... , Rj that minimize the RSS, given by

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2,$$

where ˆyrj is the mean response for the training observations within the jth box. Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes. For this reason, we take a top-down, greedy approach that is known as recursive binary splitting. It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step. Once the regions R1,...,Rj have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.

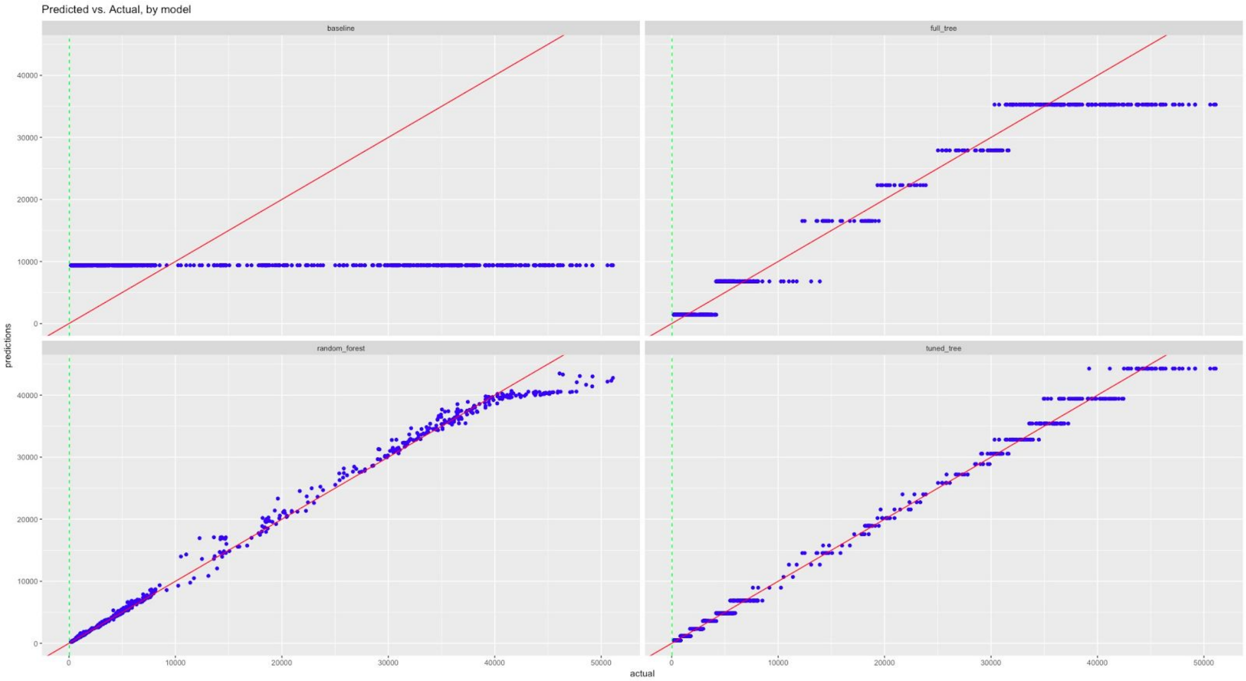
Random forests provide an improvement over decision trees as well as bagged/bootstrapped trees by way of a random small tweak that decorrelates the trees. As in bagging, we build a number forest of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose m ≈ √p — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors.

In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. This may sound crazy, but it has a clever rationale. Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, on average (p−m)/p of the splits will not even consider the strong predictor, and so other predictors will have more of a chance. We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable. *The main difference between bagging and random forests is the choice of predictor subset size m.*

2.b Results

Overall performance of all models:



As we can clearly observe in this overview, that the decision tree model uses 6 leaf nodes to predict the response of new observations. It does a pretty good job of predicting values for observations, but since the data set is huge and the response variable ranges from 0 - 70k approx., this leaves room for a lot of bias. Pruning the tree did not help since the cross-validation error for the 10 fold methods showed that the error is least on the leaves.

Also it was evident that next year's GNI depends heavily on current year's GNI.
After a little bit of playing with the tuning parameters of the function, we were able to build a model that had the least Root mean square error (RMSE) and Mean Average Error (MAE). And it is quite evident from the performance graph that it did a better job of predicting the values correctly. But the best performer of all was Random Forests. It is evident from the list of important predictors that it captured some of some predictors that were missing in decision trees.

	Method	Root mean square error	Mean average error
1	Baseline	16025.254	12581.6042
2	Full tree	3106.532	1961.5769
3	Tuned Full tree	1005.912	597.8567
4	Random Forests	1363.667	618.3239

Discussion

Random forests select a set of predictors arbitrarily each iteration to build tree, even if each one is not considered important at first. This procedure leads to a reduction in variance. Random forests can be computationally expensive, especially if we grow more trees to achieve higher accuracy. Given a representative sample, in theory there will not be any overfitting if many trees are grow. An evident improvement that can be applied is to grow more trees to increase the accuracy of the model. Even though the tuned full tree model has a lower RMSE than Random Forest, the parameters were selected using the validation set. The random forest model will likely outperform the tuned tree for new observations. As far as RNN is concerned, the validation RMSE of 705 marginally outperforms Random Forests RMSE of 1006. Possible improvements include increasing the ensemble of trees from 500 to an appropriate number for which the mean squared error is minimum. We tried that but fell short because of limited compute resources. Similar computational restrictions were present when training the RNN model. An exhaustive grid search was needed to select the model hyperparameters via cross-validation but was proven impractical due to available compute resources. These hyperparameters were finally chosen via experimentation.

Ensemble models are much more interpretable and offer more insight about the relationship between data and the future year's GNI per capita than do recurrent neural networks. Ensemble methods enabled us to conclude that the previous year's GNI per capita, GDP per capita and household final consumption expenditure are the most important predictors of the next year's GNI per capita. Interpreting a neural network the size of the one used in this analysis is difficult because the output is a complex nonlinear function of the predictors. The effect of changes in one predictor depend on the state of the rest of the predictors, making empirical analysis impractical. Therefore, the RNN model is the best choice if the goal is to generate accurate forecasts. The random forests ensemble model is the best choice of the goal is to understand the relationship between economic indicators and GNI per capita for the upcoming year.

References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2015). An introduction to statistical learning : With applications in R (Corrected at 6th printing.). (Springer texts in statistics). doi:10.1007/978-1-4614-7138-7
2. Kaggle - https://www.kaggle.com/worldbank/world-development-indicators
3. Pedro, M. (2015, April 6). Modelling – predicting the amount of rain [Web log post]. Retrieved from https://www.r-bloggers.com/
4. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (November 1997), 1735-1780. DOI=http://dx.doi.org/10.1162/neco.1997.9.8.1735
5. LSTM Architecture. Digital image. N.p., n.d. Web. 8 Apr. 2017.

Appendix

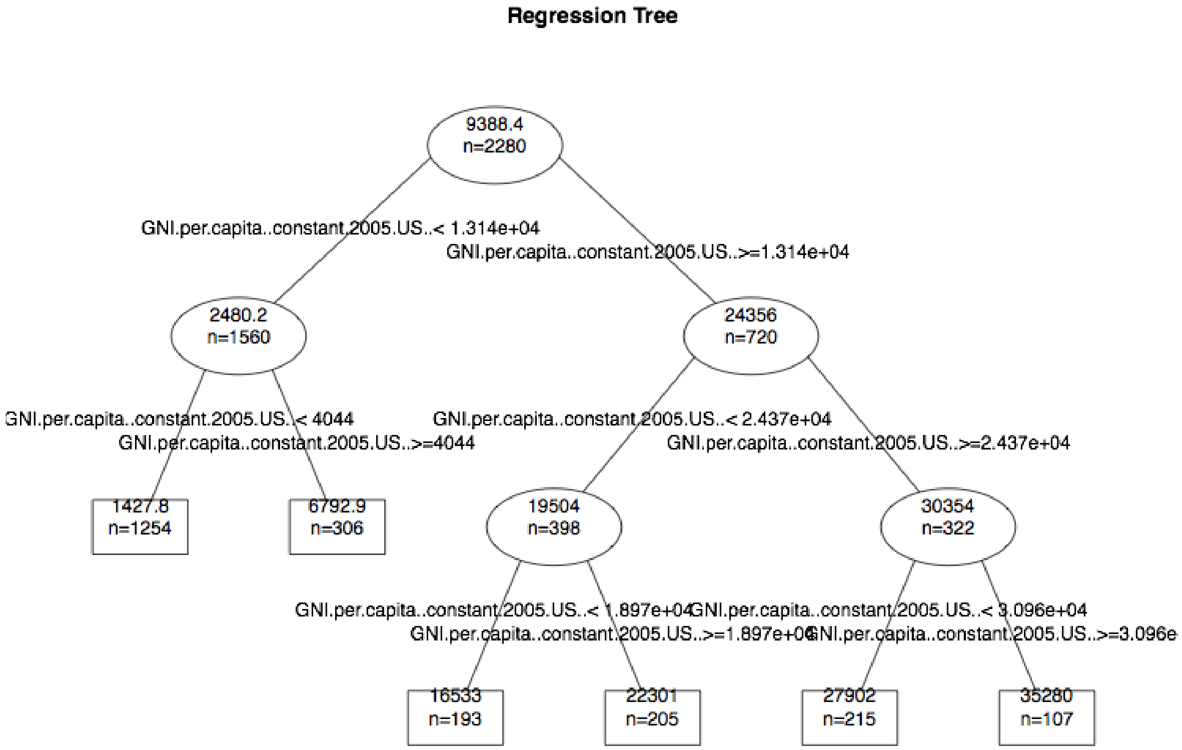
Properties of data sets:

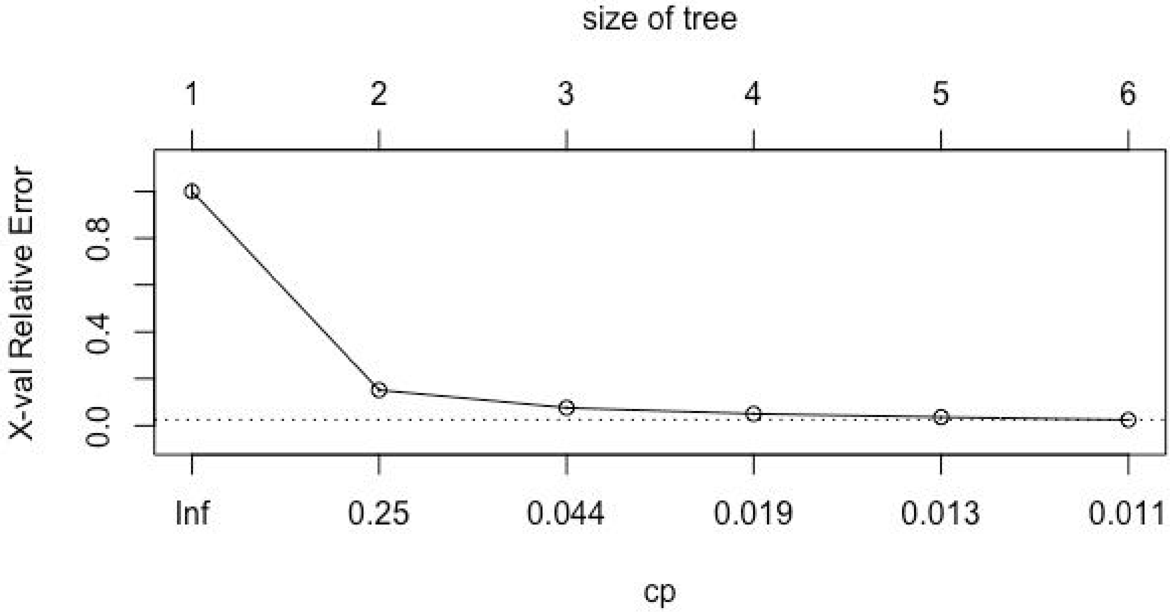
Data Set	Consecutive years	Minimum observations per predictor	Percent in validation set	Models
All nulls removed	40	5000	25	RNN
Nulls imputed	40	10	25	Random forest
Nulls preserved	40	N/A	25	Decision trees

Full tree additional results:

Summary

```
> summary(fit) # detailed summary of splits
Call:
rpart(formula = GNI.per.capita..constant.2005.US.._next ~ .,
        data = training_set, method = "anova")
  n= 2280

   CP           nsplit   rel error      xerror        xstd
1 0.84951970    0        1.00000000     1.00093640    0.027690695
2 0.07550373    1        0.15048030     0.15108625    0.006629717
3 0.02551317    2        0.07497657     0.07579695    0.003217050
4 0.01401112    3        0.04946340     0.05037436    0.002483007
5 0.01191811    4        0.03545228     0.03646629    0.001431486
6 0.01000000    5        0.02353417     0.02437688    0.001132489

Variable importance
                      GNI.per.capita..constant.2005.US..
                                          19
                      GDP.per.capita..constant.2005.US..
                                          18
                                     CountryName
                                          16
                     Birth.rate..crude..per.1.000.people.
                                          13
                  Population.ages.65.and.above....of.total.
                                          13
           Age.dependency.ratio..old....of.working.age.population.
                                          13
        Adjusted.net.national.income.per.capita..constant.2005.US..
                                          2
Household.final.consumption.expenditure.per.capita..constant.2005.US..
                                          2
                  GNI.per.capita..Atlas.method..current.US..
                                          2
```


Regression Tree
9388.4
n=2280
GNI.per.capita..constant.2005.US..< 1.314e+04
GNI.per.capita..constant.2005.US..>=1.314e+04
2480.2
n=1560
24356
n=720
GNI.per.capita..constant.2005.US..< 4044
GNI.per.capita..constant.2005.US..>=4044
GNI.per.capita..constant.2005.US..< 2.437e+04
GNI.per.capita..constant.2005.US..>=2.437e+04
1427.8
n=1254
6792.9
n=306
19504
n=398
30354
n=322
GNI.per.capita..constant.2005.US..< 1.897e+04
GNI.per.capita..constant.2005.US..>=1.897e+04
GNI.per.capita..constant.2005.US..< 3.096e+04
GNI.per.capita..constant.2005.US..>=3.096e
16533
n=193
22301
n=205
27902
n=215
35280
n=107

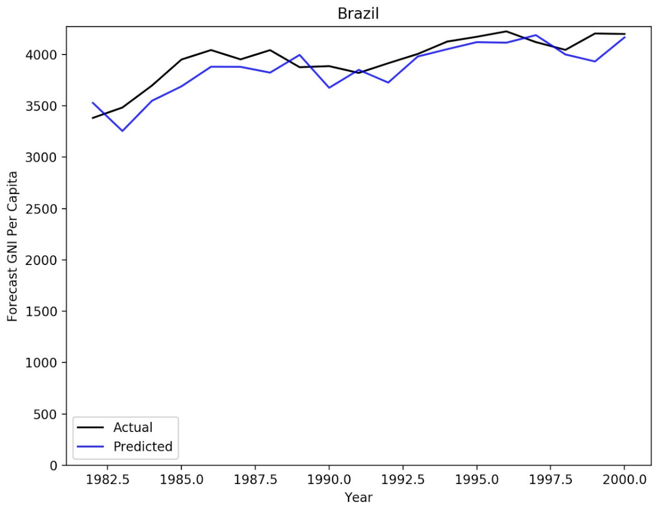

size of tree
1
2
3
4
5
6
X-val Relative Error
0.0
0.4
0.8
Inf
0.25
0.044
0.019
0.013
0.011
cp

Important predictors reported by random forests:

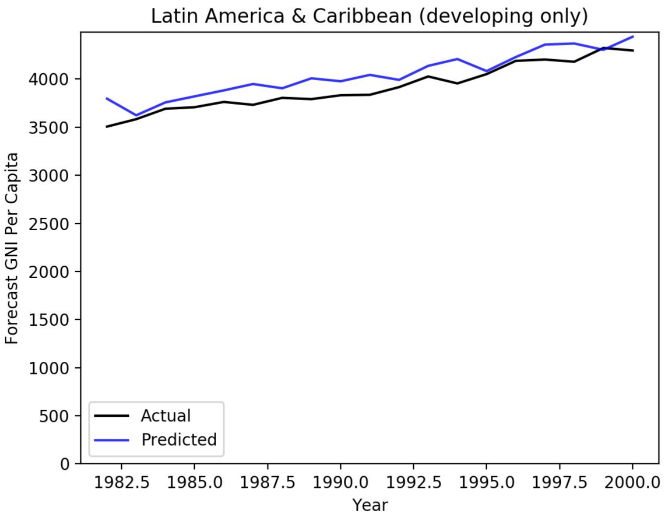
```
> head(imp_predictors)

GNI.per.capita..constant.2005.US..                                          21.416233
GDP.per.capita..constant.2005.US..                                          15.865114
Household.final.consumption.expenditure.per.capita..constant.2005.US..     11.723383
Adjusted.net.national.income.per.capita..constant.2005.US..                 9.987319
GDP.per.capita..current.US..                                                5.556497
Electric.power.consumption..kWh.per.capita.                                 4.882961
```

RNN Predicted vs. Actual per country:

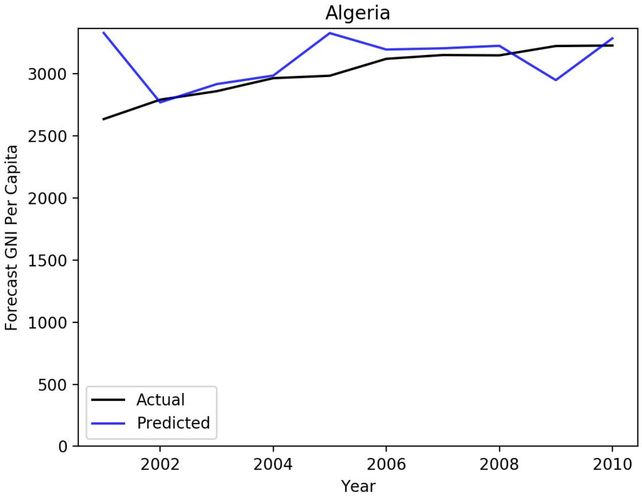
Training:



Test:



Validation:


Algeria
Forecast GNI Per Capita
0
500
1000
1500
2000
2500
3000
2002
2004
2006
2008
2010
Year
Actual
Predicted