Applications of Random Forest Algorithm

Rosie Zou[1] Matthias Schonlau, Ph.D.[2]

[1]Department of Computer Science
University of Waterloo

[2]Professor, Department of Statistics
University of Waterloo

Outline

Decision Trees

- A versitile way of solving both regression and classification problems
- Non-parametric model that involves **recursively partitioning** dataset
- Partitioning criterion and stopping condition are pre-determined, usually based on **entropy**
- Entropy, or information entropy, is a representation of how much information is encoded by given data

Decision Trees

- In **information theory**, we are interested in the **quantity** of information, which can be measured by the length of the **binary representation** of given data points and values
- Hence it is intuitive to determine when/how to split a decision tree based on whether the split maximizes information gain
- At each node of a decision tree, the entropy is given by the formula:

$$E = -\sum_{i=1}^{c} p_i \times log(p_i)$$

where p_i represents the proportion of observations with class label i, $i = \{1 \cdots c\}$

- The information gain of a split can be measured by:

$$IG = E(parent) - \text{weighted sum of } E(child\ nodes)$$

Decision Trees

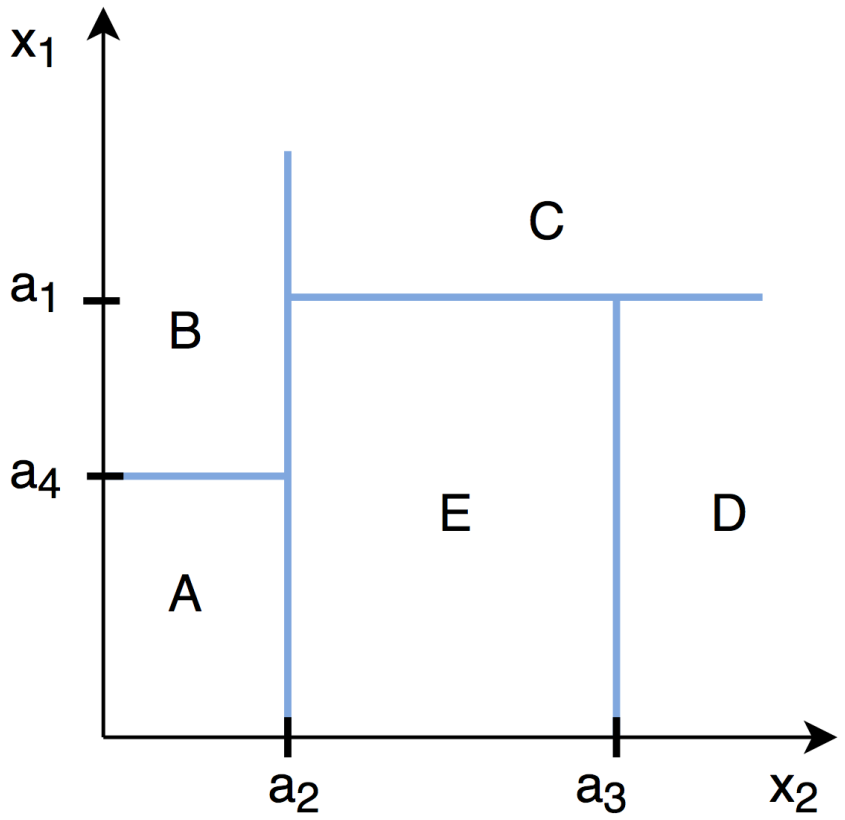

Figure: Recursive Binary Partition of a 2-Dimensional Subspace

Decision Trees

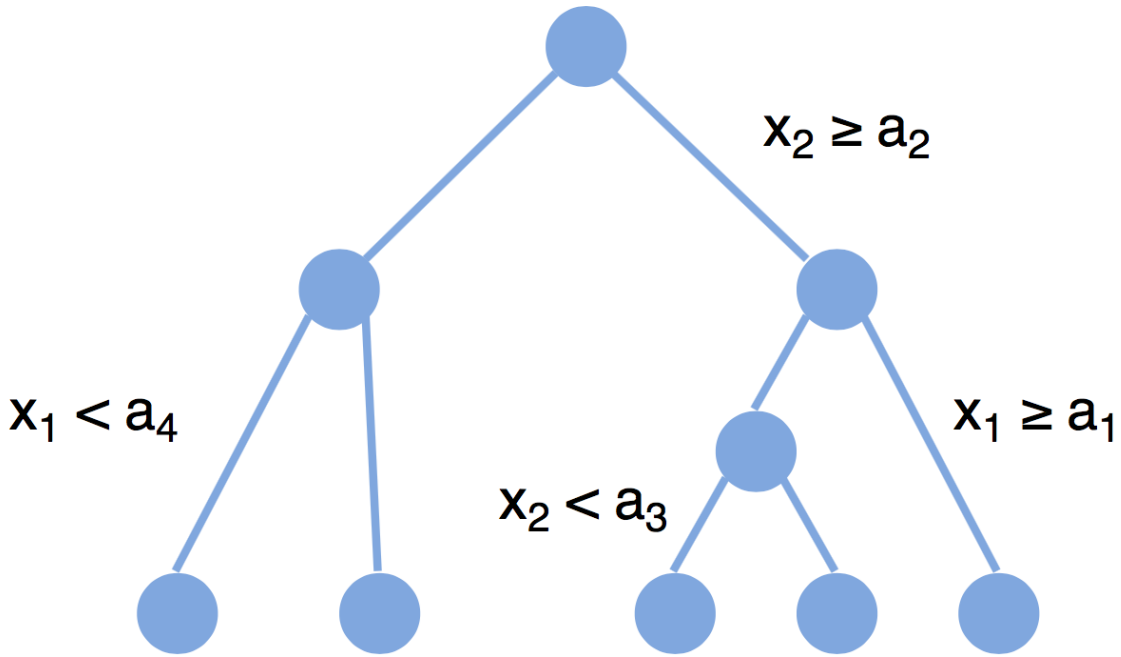

Figure: A Graphical Representation of the Decision Tree in Previous Slide

Random Forest

- A drawback of decision trees is that they are prone to over-fitting
- Over-fitting = model becomes too catered towards the training data set and performs poorly on testing data
- This will lead to low general predictive accuracy, herein referred to as generalization accuracy

Random Forest

- One way to increase generalization accuracy is to only consider a subset of the samples and build many individual trees
- Random Forest model is an **ensemble tree-based learning algorithm**; that is the algorithms averages predictions over many individual trees
- The algorithm also utilizes **bootstrap aggregating**, also known as **bagging**, to reduce overfitting and improve generalization accuracy
- Bagging refers to fitting each tree on a bootstrap sample rather than on the original sample

Random Forest

```
for i ← 1 to B by 1 do
    Draw a bootstrap sample with size N from the training data;
    while node size != minimum node size do
        randomly select a subset of m predictor variables from total p;
        for j ← 1 to m do
            if j-th predictor optimizes splitting criterion then
                split internal node into two child nodes;
                break;
            end
        end
    end
end
return the ensemble tree of all B sub-trees grown in the for loop;
```

Algorithm 1: Random Forest Algorithm

Random Forest

- Prediction for a classification problem:

$$\hat{f}(x) = \text{majority vote of all predicted classes over B trees}$$

- Prediction for a regression problem:

$$\hat{f}(x) = \text{sum of all sub-tree predictions divided over B trees}$$

Random Forest

- The random forest algorithm gives a more accurate estimate of the error rate, as compared with decision trees
- Error rate has been mathematically proven to always converge (Breiman, 2001)
- Error rate for classification problems (or root mean-squared error for regression problems) is approximated by the **out-of-bag error** during the training process
- In each tree of the random forest, the out-of-bag error is calculated based on **predictions for observations that were not in the bootstrap sample for that tree**
- Finding parameters that would produce a low out-of-bag error is often a key consideration in **parameter tuning**

Stata Syntax

- The Stata syntax to fit a random forest model is:

```
randomforest depvar indepvars [if] [in] , [ options ]
```

- Post-estimation command:

```
predict newvar | varlist | stub* [if] [in] , [ pr ]
```

Classification Example

- The following sample data comes from a 2009 research project on data mining techniques for the predictive accuracy of probability of default of credit card clients (Yeh & Lien, 2009)
- Data Description:
  - 30,000 observations, 23 variables, no missing values
  - 11 out of 23 variables are categorical
  - response variable = whether the card holder will default on his/her debt
  - predictor variables contain demographic information and banking information such as credit limit and monthly bill amount

Classification Example

First we need to randomize the data. Then we can tune the `iterations` hyper parameter.

```
set seed 201807
gen u=uniform()
sort u
gen out_of_bag_error1 = .
gen validation_error = .
gen iter1 = .
local j = 0
```

Classification Example

```
forvalues i = 10(5)500{
    local j = `j' + 1
    randomforest defaultpaymentnextmonth limit_bal sex ///
      education marriage_enum* age pay* bill* in 1/15000, ///
      type(class) iter(`i') numvars(1)
    replace iter1 = `i' in `j'
    replace out_of_bag_error1 = `e(OOB_Error)' in `j'
    predict p in 15001/30000
    replace validation_error = `e(error_rate)' in `j'
    drop p
}
```

Classification Example

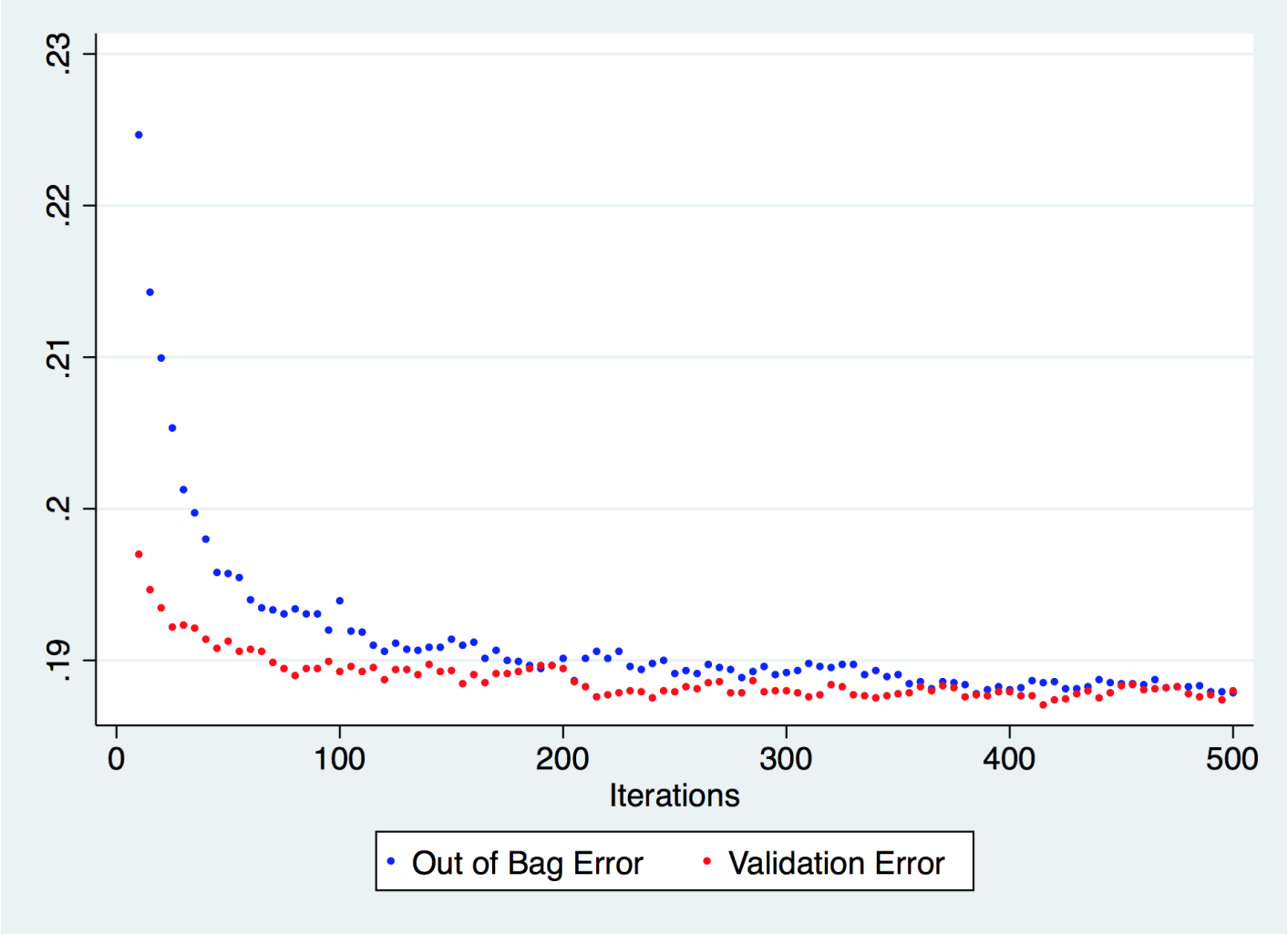

Figure: Out of Bag Error and Validation Error vs. Number of Iterations

Classification Example

Upon a first look at the graph, we can see the OOB error and validation error level off after 400 iterations and stabilize at somewhere below 0.19. Next we can tune the hyper parameter `numvars`, and fix the number of iterations to be 500.

Classification Example

```
gen oob_error = .
gen nvars = .
gen val_error = .
local j = 0
forvalues i = 1(1)26{
  local j = `j' + 1
  randomforest defaultpaymentnextmonth limit_bal sex ///
  education marriage_enum* age pay* bill* ///
  in 1/15000, type(class) iter(500) numvars(`i')
  replace nvars = `i' in `j'
  replace oob_error = `e(OOB_Error)' in `j'
  predict p in 15001/30000
  replace val_error = `e(error_rate)' in `j'
  drop p
}
```

Classification Example

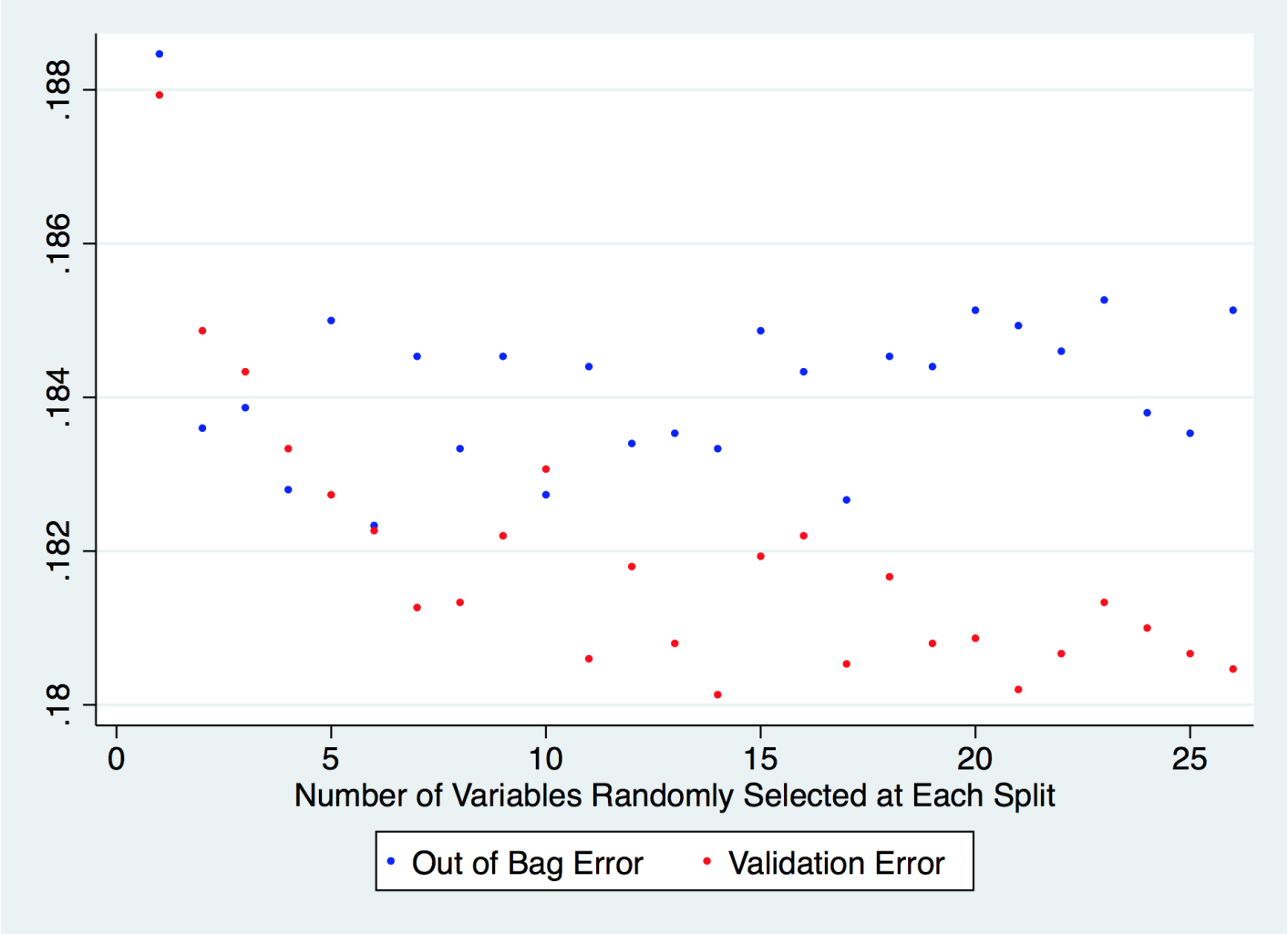

Figure: Out of Bag Error and Validation Error vs. Number of Variables

Classification Example

Finalized model is:

```
randomforest defaultpaymentnextmonth limit_bal sex ///
education marriage_enum* age pay* bill* ///
in 1/15000, type(class) iter(1000) numvars(14)
```

Classification Example

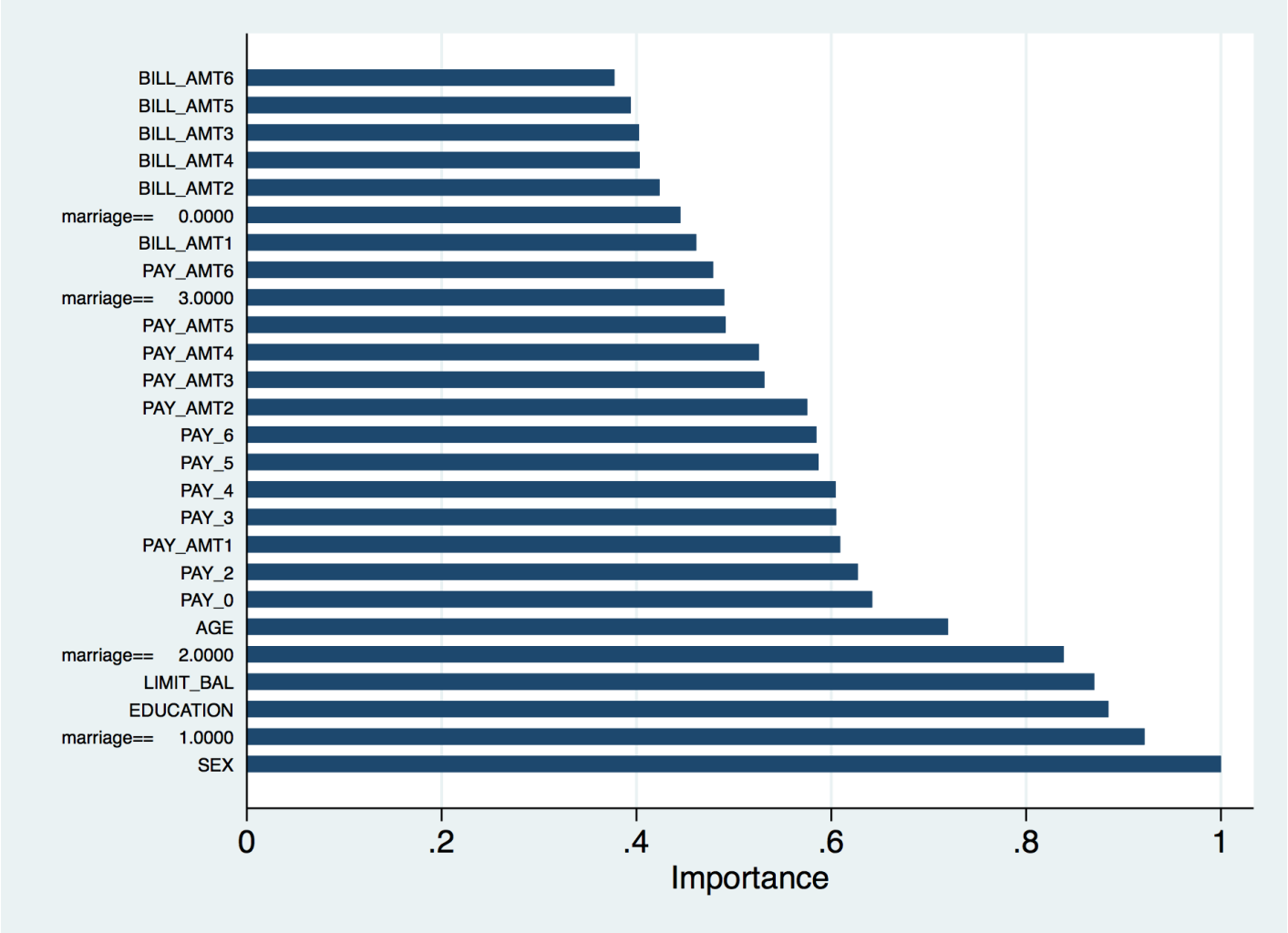

Figure: Variable Importance Plot

Classification Example

- Using the finalized model to predict against the test data (observations 15001 to 30000), the error rate is $e \approx 0.18053$
- In comparison, the testing error for a logistic regression model fitted over the same set of training data is $e = 0.1872$
- The logistic model incorrectly classifies $15000 \times (0.1872 - 0.18053) \approx 100$ observations from the test data.

Regression Example

- The following example uses data from the 2016 U.S. Consumer Finance Survey to predict log-scaled household income (Board of Governors of the Federal Reserve System, n.d.) [1]
- We will compare the root mean-squared error (RMSE) achieved by both random forest and linear regression

[1] This example is a work-in-progress and is meant for demonstration purposes only. It does not address certain data-specific issues such as multiple imputation and replicate weights.

Regression Example

First generate a separate variable for log-scaled income, then randomize the data.

```
clear
clear matrix
set maxvar 30000
use p16i6
set seed 201807
gen u = uniform()
sort u
gen logIncome = ln(X5702)
replace logIncome = 0.1 if logIncome == .
```

Regression Example

```
gen iter = .
gen validation_rmse = .
gen oob_err = .
local j = 0
forvalues i = 10(5)500{
  local j = j + 1
  randomforest logIncome X5701 X7402 X6792 X6791 ///
    X3501 X7063 X7805 X7801 X3101 X3103 X1301 ///
    X1136 X1101 X723 X501 X7973 X7126 X7122 ///
    X7593 X7592 in 1/15620, type(reg) iter(i) numvars(1)
  predict prf in 15621/31240
  replace iter = i in j
  replace validation_rmse = e(RMSE) in j
  replace oob_err = e(OOB_Error) in j
  drop prf
}
```

Regression Example

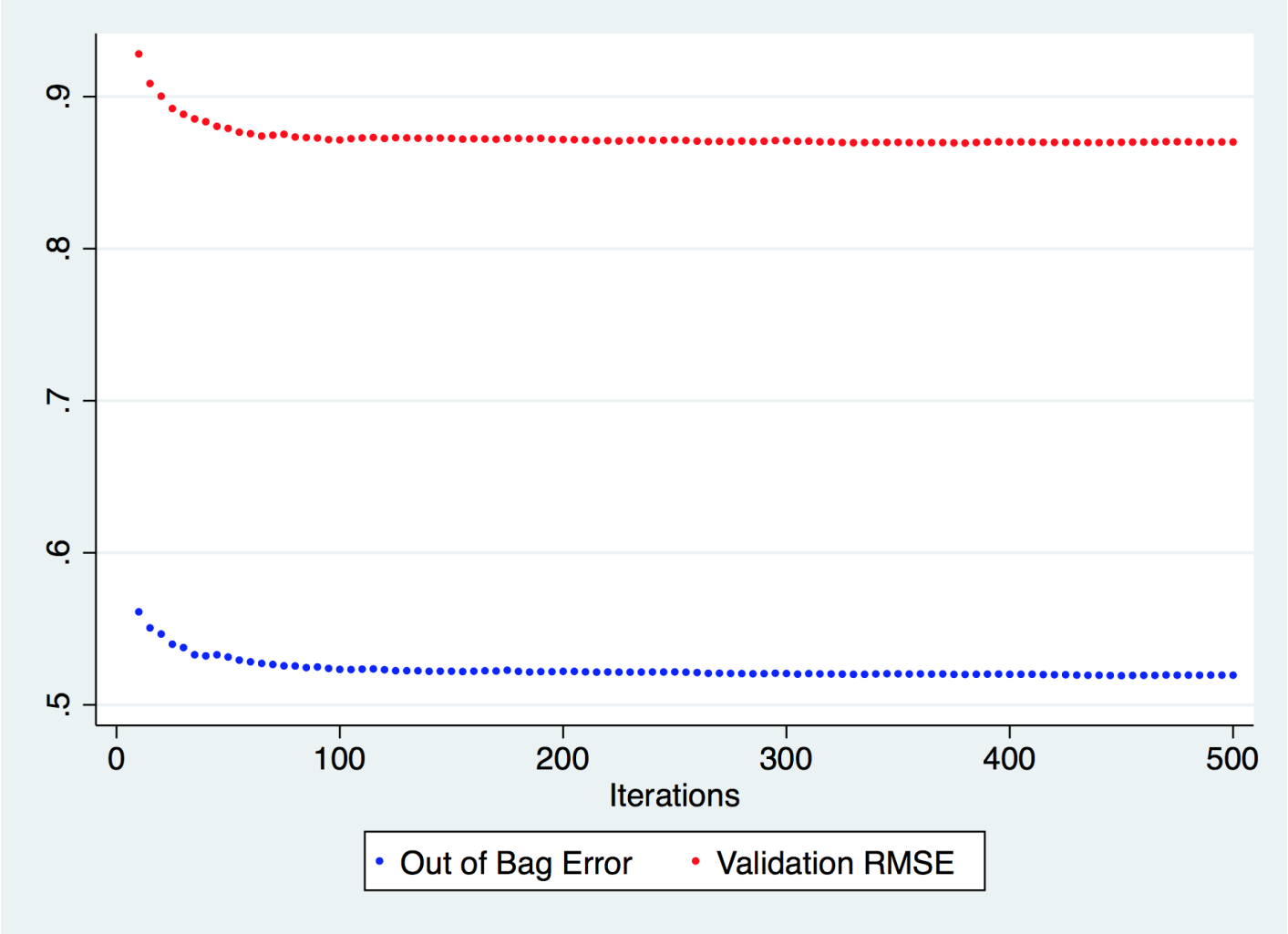

Figure: OOB Error & Validation RMSE vs. Number of Iterations

Regression Example

- We can see from the graph that the validation RMSE starts to converge to a fixed value at 100 iterations and the out-of-bag error converges after about 200 iterations
- Now we can tune the other hyper parameter, `numvars`, to see which one gives the lowest validation RMSE

Regression Example

```
gen oob_error = .
gen nvars = .
gen val_error = .
local j = 0
forvalues i = 1(1)20{
  local j = j + 1
  randomforest logIncome X5701 X7402 X6792 X6791 X3501 ///
    X7063 X7805 X7801 X3101 X3103 X1301 X1136 X1101 ///
    X723 X501 X7973 X7126 X7122 X7593 X7592 in 1/15620, ///
    type(reg) iter(500) numvars(i)
  replace nvars = i in j
  replace oob_error = e(OOB_Error) in j
  predict prf in 15621/31240
  replace val_error = e(RMSE) in j
  drop prf
}
```

Regression Example

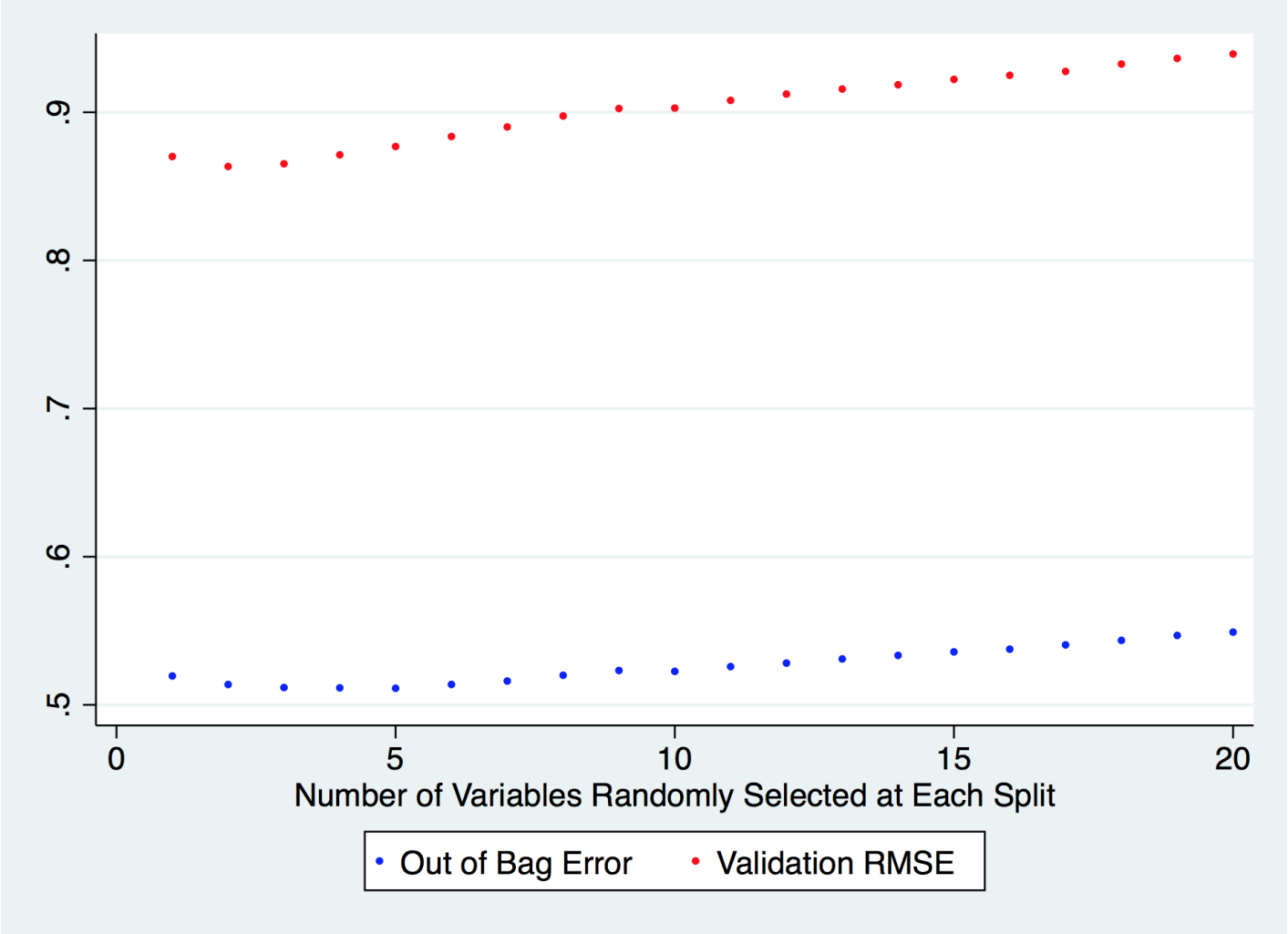

Figure: Out of Bag Error & Validation Error vs. Number of Variables

Regression Example

Finalized model is:

```
randomforest logIncome X5701 X7402 X6792 X6791 ///
X3501 X7063 X7805 X7801 X3101 X3103 X1301 X1136 ///
X1101 X723 X501 X7973 X7126 X7122 X7593 X7592 in 1/15620,
type(reg) iter(1000) numvars(2)
```

Regression Example

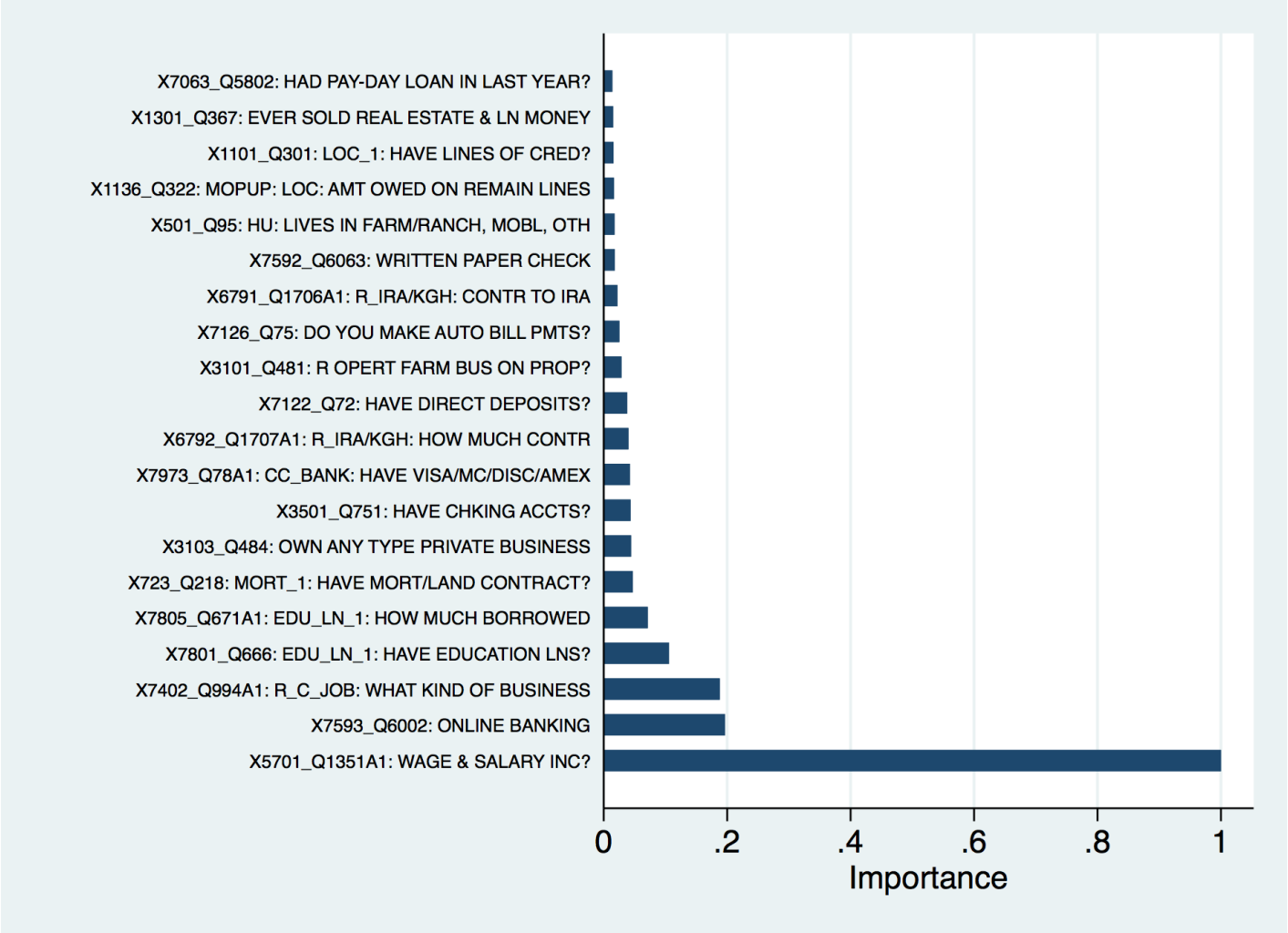

Figure: Variable Importance Plot

Regression Example

- Using the finalized model to predict against the testing data (observations 15621 to 31240), the RMSE is $e = 0.8623$
- In comparison, the testing RMSE for a linear regression model fitted over the same set of training data is $e = 1.0781$, which is 0.2158 more than the lowest error obtained from the random forest model

References and Links

- Data for classification example:
  `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`
- Data for regression example:
  `https://www.federalreserve.gov/econres/scfindex.htm`
- Original Paper on Random Forest:
  Breiman, L. (2001). Random forests. Machine Learning, 45(1), 532.