Gov 2018 Problem Set 2: Bayesian Additive Regression Trees (Solutions)

Your name here:

Friday February 25, 2022

Please upload answers to all questions (including computational questions and any related graphics and R code *with comments*, in .rmd *and* knitted pdf forms to Gradescope by Friday March 4, 2022 at 11:59 pm. For any problems that require calculations, please show your work. Finally, when doing simulations or draws from a random distribution in `R`, please set your seed immediately using `set.seed(2022)`.

Introduction

This problem is based off of Montgomery, Jacob M., Santiago Olivella, Joshua D. Potter, and Brian F. Crisp. 2015. "An Informed Forensics Approach to Detecting Vote Irregularities". *Political Analysis*.

This paper proposes an *informed forensics* approach to identifying fraud in a large cross-national setting.

- Data: a cross-national data set spanning seventy countries and six decades containing three sets of variables.
  1. Outcome (Y_i): a proxy measure of electoral fraud constructed with the NELDA data set (Hyde and Marinov 2012) using Item Response Theory (IRT) model
  2. Predictor set X_i^f: *forensic indicators* of anomalous vote distributions
  3. Predictor set X_i^c: *contextual risk factors* that have been identified in past research as increasing the likelihood of fraud.
- Method: The authors fit a Bayesian additive regression trees (BART) model developed by Chipman, George and McCulloch (2010) to combine forensic indicators and contextual risk factors.

Overview of data

	Name(s)	Description
Descriptive	`uid`	Unique id for observation $i \in \{1, \ldots, 598\}$.
	`country`	Name of country of observation i.
	`year`	Year of observation i.
Outcome	`fraud.score`	A proxy measure of electoral fraud (continuous).
Forensic indicators	`Uniform.LastDigit`	Test statistic of relative frequencies of numbers in the last digit (uniform violation).
	`DistanceLastPair`	Distance of last two digits.
	`Mean.SecondDigit`	Test statistic of relative frequencies of numbers in second digit (mean).
	`Benford.SecondDigit`	Second digit (Benford violation).
Contextual risk factors	`inequality.(var-name)`	Economic inequality (Gini coefficient). Indicator variables for each of quartiles {[20.1,27.3], (27.3,32.7], (32.7,43], (43,60.1]} and `NA`.

Name(s)	Description
`regime.(var-name)`	Average score from prior election, discretized into regime type. Indicator variables for each of {New Democracy, Old Democracy, Anocracy, Autocracy} and `NA`.
`avgmag.(var-name)`	Mean of all districts' magnitudes. Indicator variables for each of quartiles {[1,1.03], (1.03,6.17], (6.17,11.1], (11.1,150]} and `NA`.
`gdpchange.(var-name)`	Change in GDP per capita in the prior year. Indicator variables for each of quartiles {[-32.1,1.81], (1.81,3.56], (3.56,5.58], (5.58,34.8]} and `NA`.
`commision.(var-name)`	Level of independence of electoral commission (government/mixed/independent). Indicator variables for each of {0, 1, 2}.
`instability.0`	Indicator variable for political instability.
`fract`	Fearon's index of ethnic fractionalization.
`urban`	Percent of population living in urban centers.
`turnout`	Percent registered voters casting ballots in national election.

Note that the descriptive variables are not included in the model. See Table B1 of Appendix B for more details of each variable.

A quick review of ensemble methods for trees

Recall that there exist various methods to combine a set of tree models (*ensemble methods*), to produce a powerful committee using the outputs of many weak classifiers.

- Boosting: fit a sequence of single trees, using each tree to fit data variation not explained by earlier trees in the sequence, and then combine these trees to correct bias.
- Bagging (= bootstrap aggregation): fit a large number of independent trees for each of bootstrap samples, and average these noisy trees to reduce variance.
- Random forests: a modification of bagging that averages a large collection of de-correlated trees created with a random selection of the input variables.

In this problem set, we will use a Bayesian additive regression trees (BART) model, a sum-of-trees model motivated by a boosting algorithm. BART is a Bayesian approach for a sum of trees where each tree explains a small and different portion by imposing a regularization prior (see Chipman, George and McCulloch 2010 for more details). According to the authors, "one crucial advantage of using BART is that Bayesian estimation techniques allow the model to produce measures of uncertainty regarding not only the model's parameters (which are often not of direct interest), but also of more usual quantities of interest, such as the partial dependence of the outcome of interest across any one combination of covariates. (Montgomery et al. 2010, Appendix F)'' In light of this advantage, the authors answer the following question along with its uncertainty (e.g., 80% credible intervals): which particular variables are most important in allowing the fitted BART model to identify electoral fraud (as captured by our NELDA-based proxy)?

Throughout the questions, we will use the data from `forensic.RData`, and a set of tuning parameters that the original paper has used. Note that your result may look somewhat different from the results from the paper due to the random split of training and test data and the use of a different package — the authors used `BayesTree` package for their analysis while we will use `BART`, a recently published package that is relatively computationally efficient (see this link for more details). **Bonus points for efforts to tune the parameters for better performance.**

Question 1

In this question, we will fit a continuous BART model with forensic indicators.

(a) Load the data, set the random seed and randomly split the data into two sets: a training set with 500 observations, and test set with 98 observations.

```
# load data
load("forensic.RData")

# set the seed number
seed.num = 2022

# create train and test data - although I need to randomize because this is temporal data I cannot comp

train <- data %>%
  filter(year < 2001)

test <- data %>%
  filter(year > 2001)

# the year 2001 is in the middle so I have to randomize it and assign the different groups to training
y2001 <- data %>%
  filter(year==2001)

mix <-  y2001[sample(1:nrow(y2001)), ]

train01 <- mix %>% slice(1:6)

test01 <- mix %>% slice(7:16)

# bind together with randomized 2001 data
test <- rbind(test, test01)
train <- rbind(train, train01)
```

(b) Using `fraud.score` as the outcome variable, and forensic indicators as predictors (a total 4 variables), fit the BART model with your training set using a call to `wbart()`.

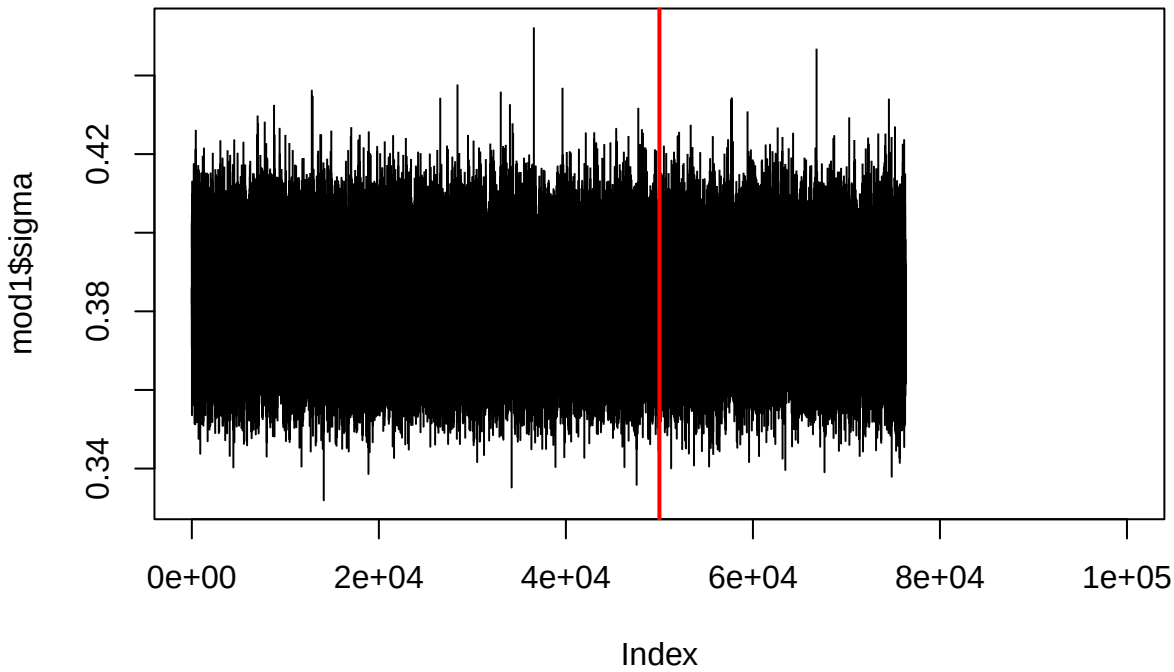
```
set.seed(seed.num)

# create x variables for train and test
x.train <- train %>% select(Uniform.LastDigit, DistanceLastPair, Mean.SecondDigit, Benford.SecondDigit)
x.test <- test %>% select(Uniform.LastDigit, DistanceLastPair, Mean.SecondDigit, Benford.SecondDigit)

# y.train has to be in vector form
y.train <- train$fraud.score
y.test <- test$fraud.score

# run model
mod1 <-  wbart(x.train = x.train,
                           y.train = y.train,
                           x.test = x.test,
                           k = 2, power = 0.5, base = 0.95, # parameters from the paper
                           sigdf = 10, sigquant = 0.75, nskip = 50000,
                           ndpost = 5000, ntree = 100, sigest = 0.35,
                           keepevery = 10, printevery = 2000)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 500, 4, 98
## y1,yn: -0.120037, 0.639549
## x1,x[n*p]: 0.076870, 0.006431
## xp1,xp[np*p]: 0.087148, 0.013015
## *****Number of Trees: 100
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 50000, 50000
## *****Prior:beta,alpha,tau,nu,lambda: 0.500000,0.950000,0.062275,10.000000,0.082531
## *****sigma: 0.350000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,4,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 5000,5000,5000,5000
## *****printevery: 2000
## *****skiptr,skipte,skipteme,skiptreedraws: 10,10,10,10
##
## MCMC
## done 0 (out of 100000)
## done 2000 (out of 100000)
## done 4000 (out of 100000)
## done 6000 (out of 100000)
## done 8000 (out of 100000)
## done 10000 (out of 100000)
## done 12000 (out of 100000)
## done 14000 (out of 100000)
## done 16000 (out of 100000)
## done 18000 (out of 100000)
## done 20000 (out of 100000)
## done 22000 (out of 100000)
## done 24000 (out of 100000)
## done 26000 (out of 100000)
## done 28000 (out of 100000)
## done 30000 (out of 100000)
## done 32000 (out of 100000)
## done 34000 (out of 100000)
## done 36000 (out of 100000)
## done 38000 (out of 100000)
## done 40000 (out of 100000)
## done 42000 (out of 100000)
## done 44000 (out of 100000)
## done 46000 (out of 100000)
## done 48000 (out of 100000)
## done 50000 (out of 100000)
## done 52000 (out of 100000)
## done 54000 (out of 100000)
## done 56000 (out of 100000)
## done 58000 (out of 100000)
## done 60000 (out of 100000)
## done 62000 (out of 100000)
## done 64000 (out of 100000)
## done 66000 (out of 100000)
```

```
## done 68000 (out of 100000)
## done 70000 (out of 100000)
## done 72000 (out of 100000)
## done 74000 (out of 100000)
## done 76000 (out of 100000)
## done 78000 (out of 100000)
## done 80000 (out of 100000)
## done 82000 (out of 100000)
## done 84000 (out of 100000)
## done 86000 (out of 100000)
## done 88000 (out of 100000)
## done 90000 (out of 100000)
## done 92000 (out of 100000)
## done 94000 (out of 100000)
## done 96000 (out of 100000)
## done 98000 (out of 100000)
## time: 121s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 5000,5000,5000,5000
```

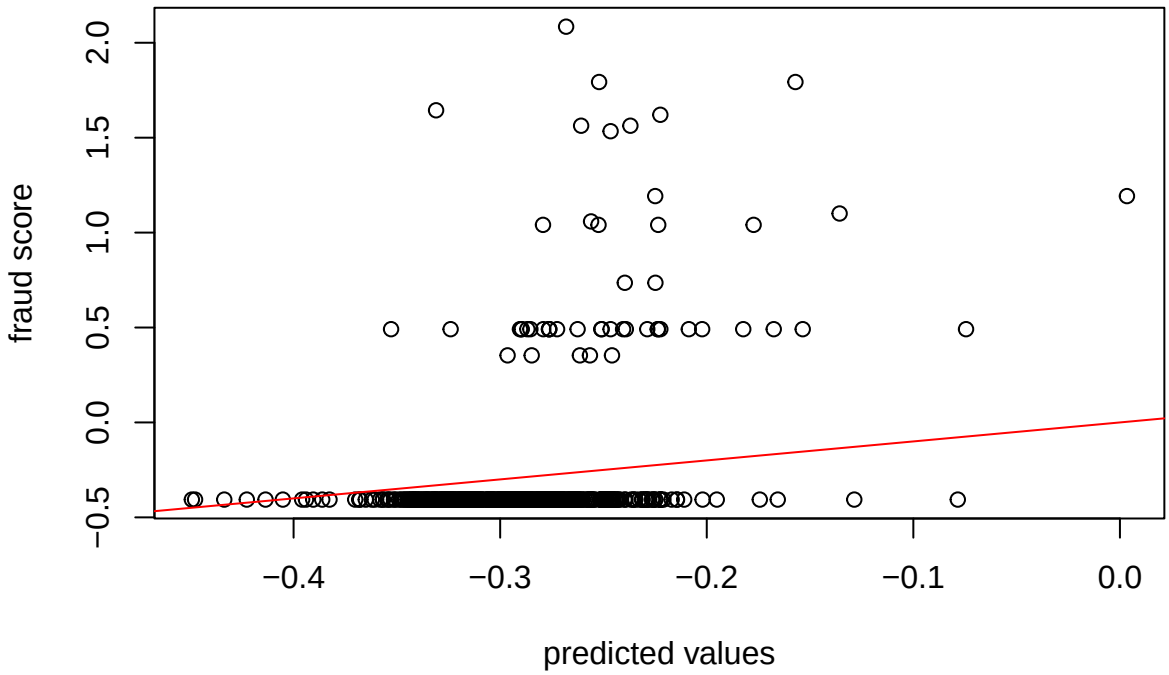
(c) Check the convergence with `plot(your-fitted-model$sigma, type = "l")`.

```
plot(mod1$sigma, type = "l")
abline(v = 50000, lwd = 2, col = "red")
```



(d) Create a scatter plot of your test set where the y-axis is the outcome variable (`fraud.score`) and the x-axis is its predictions using the fitted model. Add a 45 degree line as a reference.

```
plot(x = mod1$yhat.train.mean, y = y.train, xlab = "predicted values", ylab = "fraud score")
 abline(0, 1, col = "red")
```



(e) Report the following statistics (RMSE, MAE, MAPE, MAD, MEAPE) of the fitted model with test set.

Let $e_i = \mid \hat{y}_i - y_i \mid$ denote the absolute error, and $a_i = \frac{e_i}{|y_i|} \times 100$ denote the absolute percentage error where y_i is the true outcome value and $\hat{y}_i$ is the prediction.

- Root mean squared error (RMSE) $= \sqrt{\frac{\sum_i e_i^2}{n}}$
- Mean absolute error (MAE) $= \frac{\sum_i e_i}{n}$
- Mean absolute proportional error (MAPE) $= \frac{\sum_i a_i}{n}$
- Median absolute deviation (MAD) $= \text{med}(\mathbf{e})$
- Median absolute proportional error (MEAPE) $= \text{med}(\mathbf{a})$

Note that n here would be the number of observations in the test set ($= 98$), $\mathbf{e} = (e_1, \ldots, e_n)$, and $\mathbf{a} = (a_1, \ldots, a_n)$

```
## (e)
# Create a function for computing the test statistics
stats.model <- function(true.y, predict.y) {
  if (length(true.y)!=length(predict.y)) stop("Check the inputs.")
  # TODO: specify the following objects
  n = length(true.y)
  e = abs(predict.y - true.y)
  a = (e/(abs(true.y)))*100
  rmse = sqrt(sum(e^2)/n)
  mae = sum(e)/n
  mape = sum(a)/n
  mad = median(e)
  meape = median(a)
  res = c(rmse, mae, mape, mad, meape)
  names(res) = c("rmse", "mae", "mape", "mad", "meape")
  return(res)
}

# TODO: specify the arguments
```

```r
stats.model(test$fraud.score, mod1$yhat.test.mean)
```

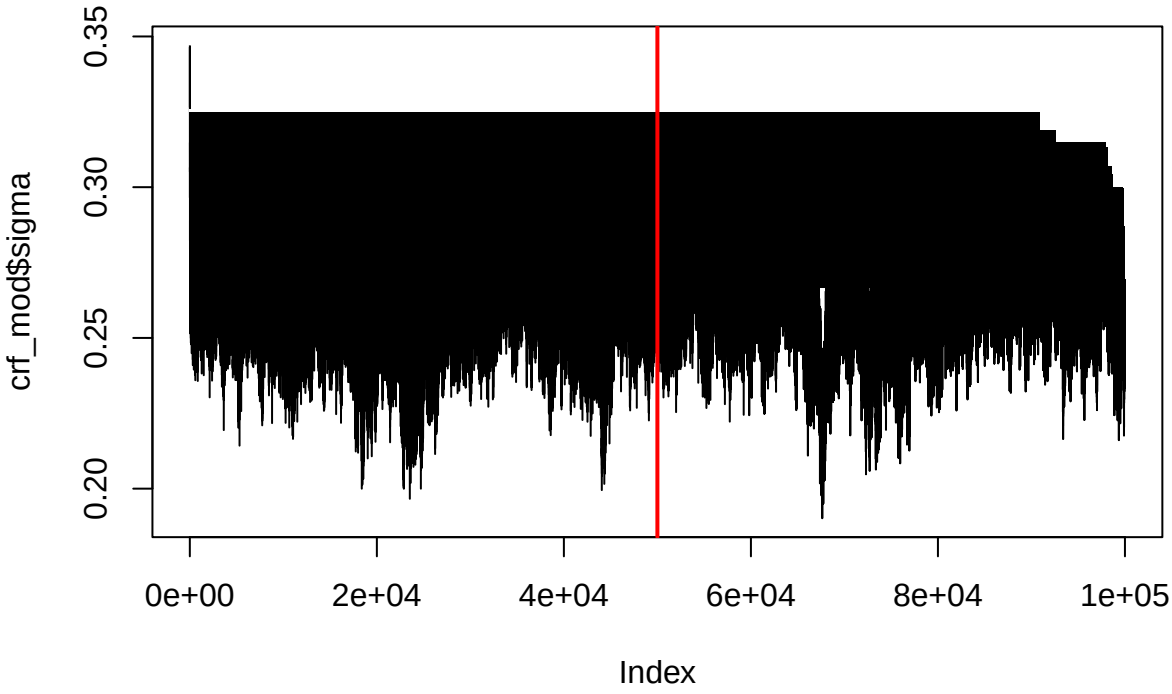
```
##       rmse        mae       mape        mad      meape
##  0.6330435  0.3706779 56.8376437  0.1366950 33.6466208
```

```r
##### CONTEXTUAL RISK FACTORS############

# because these models include different numbers of variables I need to recreate x.train and x.test

# deselect the informed forensics and other variables
x.train.crf <- train %>%
  select(!c(Uniform.LastDigit, DistanceLastPair, Mean.SecondDigit, Benford.SecondDigit, fraud.score, ui
x.test.crf <- test %>%
  select(!c(Uniform.LastDigit, DistanceLastPair, Mean.SecondDigit, Benford.SecondDigit, fraud.score, ui


# use the same code as before

set.seed(seed.num)
# TODO: specify the arguments (x.train, y.train, x.test)
crf_mod <- wbart(x.train = x.train.crf,
                           y.train = y.train,
                           x.test = x.test.crf,
                           k = 2, power = 0.5, base = 0.95, # parameters from the paper
                           sigdf = 10, sigquant = 0.75, nskip = 50000,
                           ndpost = 5000, ntree = 100, sigest = 0.35,
                           keepevery = 10, printevery = 2000)
```

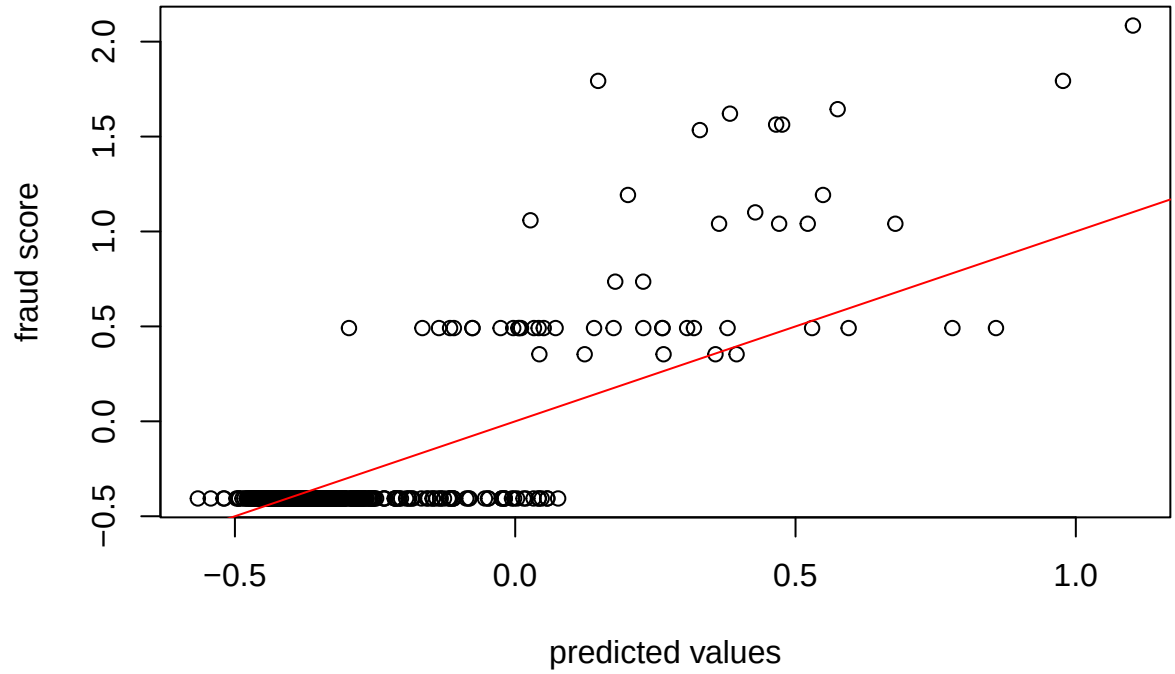
```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 500, 27, 98
## y1,yn: -0.120037, 0.639549
## x1,x[n*p]: 0.126000, 0.000000
## xp1,xp[np*p]: 0.647000, 0.000000
## *****Number of Trees: 100
## *****Number of Cut Points: 62 ... 1
## *****burn and ndpost: 50000, 50000
## *****Prior:beta,alpha,tau,nu,lambda: 0.500000,0.950000,0.062275,10.000000,0.082531
## *****sigma: 0.350000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,27,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 5000,5000,5000,5000
## *****printevery: 2000
## *****skiptr,skipte,skipteme,skiptreedraws: 10,10,10,10
##
## MCMC
## done 0 (out of 100000)
## done 2000 (out of 100000)
## done 4000 (out of 100000)
## done 6000 (out of 100000)
## done 8000 (out of 100000)
## done 10000 (out of 100000)
## done 12000 (out of 100000)
## done 14000 (out of 100000)
```

```
## done 16000 (out of 100000)
## done 18000 (out of 100000)
## done 20000 (out of 100000)
## done 22000 (out of 100000)
## done 24000 (out of 100000)
## done 26000 (out of 100000)
## done 28000 (out of 100000)
## done 30000 (out of 100000)
## done 32000 (out of 100000)
## done 34000 (out of 100000)
## done 36000 (out of 100000)
## done 38000 (out of 100000)
## done 40000 (out of 100000)
## done 42000 (out of 100000)
## done 44000 (out of 100000)
## done 46000 (out of 100000)
## done 48000 (out of 100000)
## done 50000 (out of 100000)
## done 52000 (out of 100000)
## done 54000 (out of 100000)
## done 56000 (out of 100000)
## done 58000 (out of 100000)
## done 60000 (out of 100000)
## done 62000 (out of 100000)
## done 64000 (out of 100000)
## done 66000 (out of 100000)
## done 68000 (out of 100000)
## done 70000 (out of 100000)
## done 72000 (out of 100000)
## done 74000 (out of 100000)
## done 76000 (out of 100000)
## done 78000 (out of 100000)
## done 80000 (out of 100000)
## done 82000 (out of 100000)
## done 84000 (out of 100000)
## done 86000 (out of 100000)
## done 88000 (out of 100000)
## done 90000 (out of 100000)
## done 92000 (out of 100000)
## done 94000 (out of 100000)
## done 96000 (out of 100000)
## done 98000 (out of 100000)
## time: 193s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 5000,5000,5000,5000
```

```r
## (c)
# TODO: specify your model
plot(crf_mod$sigma, type = "l")
abline(v = 50000, lwd = 2, col = "red")
```




```
## (d)
# TODO: create a scatter plot
# same code as above
plot(x = crf_mod$yhat.train.mean, y = y.train, xlab = "predicted values", ylab = "fraud score")
 abline(0, 1, col = "red")
```




```
## (e)
# test statistic model has already been created
# input new model into stats.model function
stats.model(test$fraud.score, crf_mod$yhat.test.mean)
```

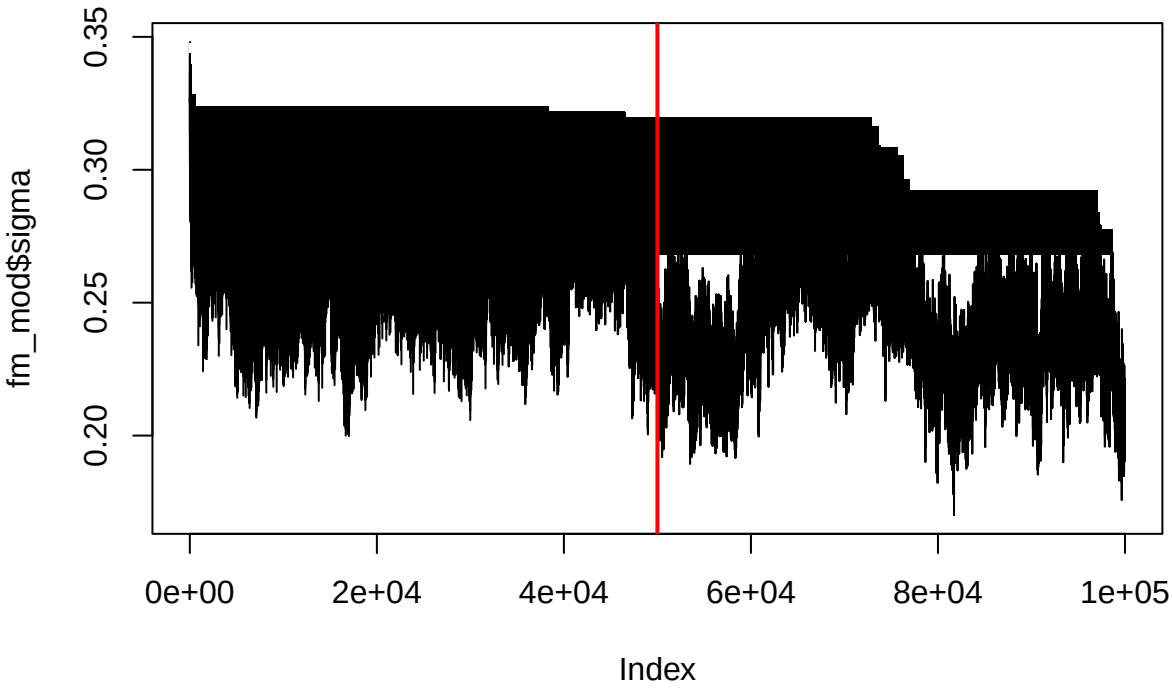
```
##       rmse        mae       mape        mad      meape
##  0.5787125  0.3581729 59.1907272  0.1907810 46.7339784
```

```
##### FORENSICS MODEL ############

# because these models include different numbers of variables I need to recreate x.train and x.test

# deselect the other variables
x.train.fm <- train %>%
  select(!c(fraud.score, uid, country, year))
x.test.fm <- test %>%
  select(!c(fraud.score, uid, country, year))


# use the same code as before

set.seed(seed.num)
# TODO: specify the arguments (x.train, y.train, x.test)
fm_mod <- wbart(x.train = x.train.fm,
                            y.train = y.train,
                            x.test = x.test.fm,
                            k = 2, power = 0.5, base = 0.95, # parameters from the paper
                            sigdf = 10, sigquant = 0.75, nskip = 50000,
                            ndpost = 5000, ntree = 100, sigest = 0.35,
                            keepevery = 10, printevery = 2000)
```

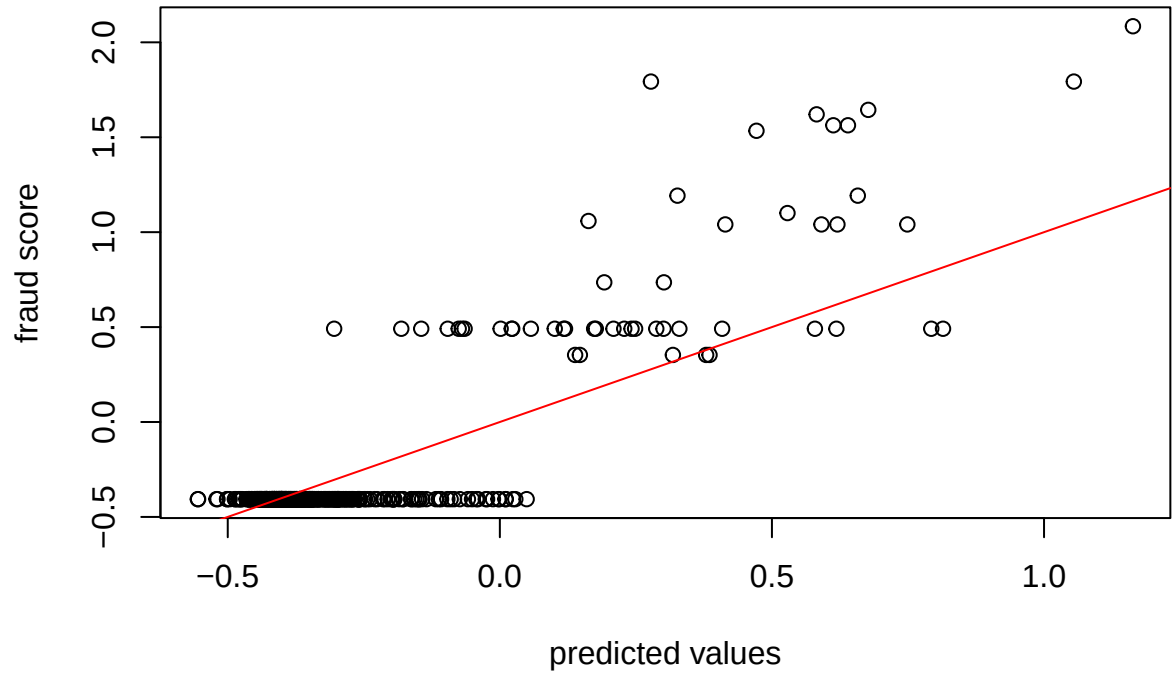
```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 500, 31, 98
## y1,yn: -0.120037, 0.639549
## x1,x[n*p]: 0.076870, 0.000000
## xp1,xp[np*p]: 0.087148, 0.000000
## *****Number of Trees: 100
## *****Number of Cut Points: 100 ... 1
## *****burn and ndpost: 50000, 50000
## *****Prior:beta,alpha,tau,nu,lambda: 0.500000,0.950000,0.062275,10.000000,0.082531
## *****sigma: 0.350000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,31,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 5000,5000,5000,5000
## *****printevery: 2000
## *****skiptr,skipte,skipteme,skiptreedraws: 10,10,10,10
##
## MCMC
## done 0 (out of 100000)
## done 2000 (out of 100000)
## done 4000 (out of 100000)
## done 6000 (out of 100000)
## done 8000 (out of 100000)
## done 10000 (out of 100000)
## done 12000 (out of 100000)
## done 14000 (out of 100000)
## done 16000 (out of 100000)
## done 18000 (out of 100000)
## done 20000 (out of 100000)
```

```
## done 22000 (out of 100000)
## done 24000 (out of 100000)
## done 26000 (out of 100000)
## done 28000 (out of 100000)
## done 30000 (out of 100000)
## done 32000 (out of 100000)
## done 34000 (out of 100000)
## done 36000 (out of 100000)
## done 38000 (out of 100000)
## done 40000 (out of 100000)
## done 42000 (out of 100000)
## done 44000 (out of 100000)
## done 46000 (out of 100000)
## done 48000 (out of 100000)
## done 50000 (out of 100000)
## done 52000 (out of 100000)
## done 54000 (out of 100000)
## done 56000 (out of 100000)
## done 58000 (out of 100000)
## done 60000 (out of 100000)
## done 62000 (out of 100000)
## done 64000 (out of 100000)
## done 66000 (out of 100000)
## done 68000 (out of 100000)
## done 70000 (out of 100000)
## done 72000 (out of 100000)
## done 74000 (out of 100000)
## done 76000 (out of 100000)
## done 78000 (out of 100000)
## done 80000 (out of 100000)
## done 82000 (out of 100000)
## done 84000 (out of 100000)
## done 86000 (out of 100000)
## done 88000 (out of 100000)
## done 90000 (out of 100000)
## done 92000 (out of 100000)
## done 94000 (out of 100000)
## done 96000 (out of 100000)
## done 98000 (out of 100000)
## time: 185s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 5000,5000,5000,5000
```

```r
## (c)
# TODO: specify your model
plot(fm_mod$sigma, type = "l")
abline(v = 50000, lwd = 2, col = "red")
```




```
## (d)
# TODO: create a scatter plot
# same code as above
plot(x = fm_mod$yhat.train.mean, y = y.train, xlab = "predicted values", ylab = "fraud score")
 abline(0, 1, col = "red")
```




```
## (e)
# test statistic model has already been created
# input new model into stats.model function
stats.model(test$fraud.score, fm_mod$yhat.test.mean)
```

```
##       rmse        mae       mape        mad      meape
##  0.5923762  0.3634816 60.2025199  0.1600596 39.6376776
```

```
# run stats model for all three models together so I can compare outcomes better

stats.model(test$fraud.score, mod1$yhat.test.mean)
```

```
##       rmse        mae       mape        mad      meape
##  0.6330435  0.3706779 56.8376437  0.1366950 33.6466208
```

```
stats.model(test$fraud.score, crf_mod$yhat.test.mean)
```

```
##       rmse        mae       mape        mad      meape
##  0.5787125  0.3581729 59.1907272  0.1907810 46.7339784
```

```
stats.model(test$fraud.score, fm_mod$yhat.test.mean)
```

```
##       rmse        mae       mape        mad      meape
##  0.5923762  0.3634816 60.2025199  0.1600596 39.6376776
```

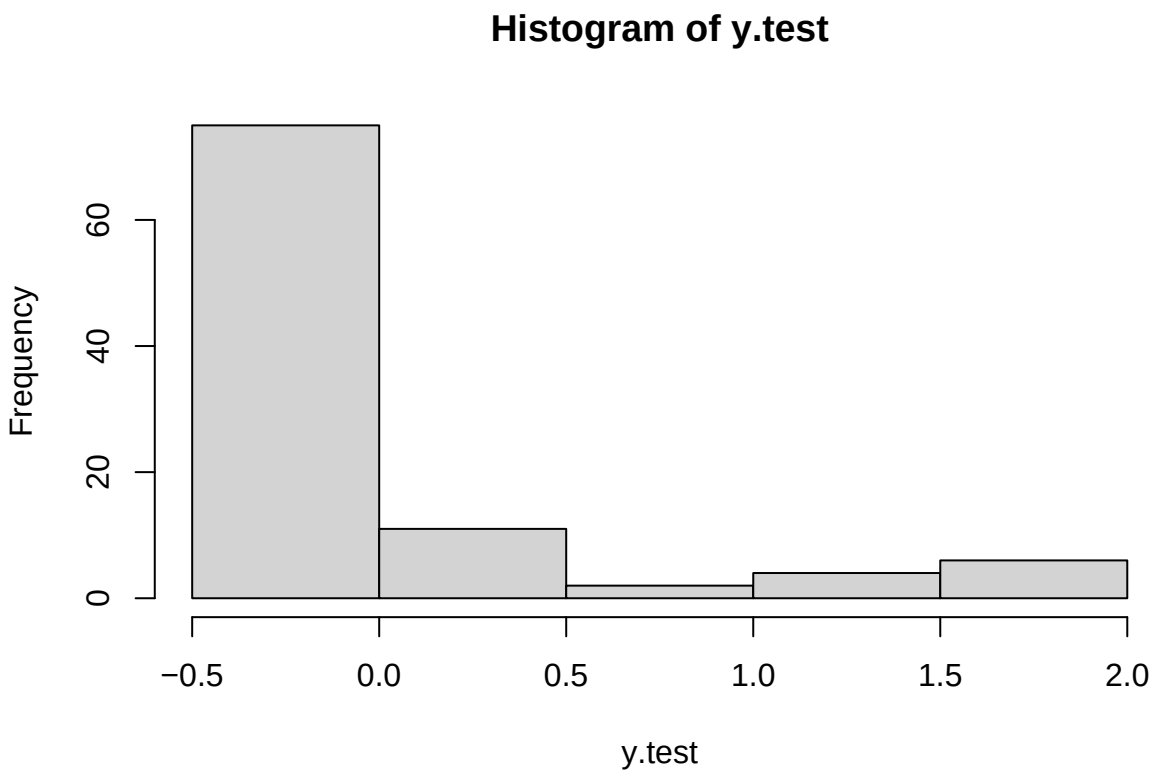
All three of the models have similar error levels. The informed forensics model has the lowest RMSE, MAE, and MAPE (the MAPE values are very similar for all three models: the forensic indicators model has the highest MAPE and the contextual risk factors model is in the middle). For MAD and MEAPE the forensic indicators model has the lowest error by a lot (the informed forensics has the highest error for MAD and MEAPE, again the contextual risk factors model is in the middle).

Question 3

Now, we will investigate a potential concern with these results: "given that roughly five out of six of our observations take on the lowest fraud score, these fit statistics may simply be capturing the models' accurate prediction of cases with no obvious fraudulent activities (496).'' (A case of *high specificity but low sensitivity*)

(a) Create a histogram of the outcome variable of your test data and check the distribution. Specifically, how many observations in your test data have the lowest fraud score ($= -0.4062667$)?

```
# create a histogram
hist(y.test)
```




```
# find exact number for the lowest level - filter didn't work for some reason
data.frame(y.test) %>%
  group_by(y.test) %>%
  mutate(count = n()) %>%
  distinct()
```

```
## # A tibble: 9 x 2
## # Groups:   y.test [9]
##   y.test count
##    <dbl> <int>
## 1 -0.406    75
## 2  1.67      2
## 3  1.94      1
## 4  0.353     3
## 5  1.19      3
## 6  0.491     8
## 7  1.04      1
## 8  0.572     2
## 9  1.53      3
```

There are 74 observations in my test dataset with the lowest fraud score.

(b) Create a binary version of the outcome variable (`fraud.score`) with the threshold at 0 for both of your train and test set.

```
# put data into dfs so I can use ifelse
traindf <- data.frame(y.train) %>%
  mutate(new = ifelse(y.train > 0,1,0))
testdf <- data.frame(y.test) %>%
  mutate(new = ifelse(y.test > 0,1,0))
```

```r
# create vectors
y.train.bin <- traindf$new
y.test.bin <- testdf$new
```

(c) Fit a probit BART using this binary outcomes with informed forensics model (a total 31 predictors).

```r
# create new x data with all variables
x.test.if <- test %>%
  select(!c(fraud.score, uid, country, year))
x.train.if <- train %>%
  select(!c(fraud.score, uid, country, year))

# use the sample code from below

# set.seed(seed.num)
# TODO: specify the arguments (x.train, y.train, x.test)
binary.mod <- pbart(x.train = x.train.if,
                    y.train = y.train.bin,
                    x.test = x.test.if,
                    k = 2, power = 0.5, base = 0.95,
                    nskip = 50000, ndpost = 5000, ntree = 100,
                    keepevery = 10, printevery = 2000)
```

```
## *****Into main of pbart
## *****Data:
## data:n,p,np: 500, 31, 98
## y1,yn: 0, 1
## x1,x[n*p]: 0.076870, 0.000000
## xp1,xp[np*p]: 0.087148, 0.000000
## *****Number of Trees: 100
## *****Number of Cut Points: 100 ... 1
## *****burn and ndpost: 50000, 50000
## *****Prior:mybeta,alpha,tau: 0.500000,0.950000,0.150000
## *****binaryOffset: -1.270238
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,31,0
## *****nkeeptrain,nkeeptest,nkeeptreedraws: 5000,5000,5000
## *****printevery: 2000
## *****skiptr,skipte,skiptreedraws: 10,10,10
##
## MCMC
## done 0 (out of 100000)
## done 2000 (out of 100000)
## done 4000 (out of 100000)
## done 6000 (out of 100000)
## done 8000 (out of 100000)
## done 10000 (out of 100000)
## done 12000 (out of 100000)
## done 14000 (out of 100000)
## done 16000 (out of 100000)
## done 18000 (out of 100000)
## done 20000 (out of 100000)
## done 22000 (out of 100000)
## done 24000 (out of 100000)
## done 26000 (out of 100000)
## done 28000 (out of 100000)
```

```
## done 30000 (out of 100000)
## done 32000 (out of 100000)
## done 34000 (out of 100000)
## done 36000 (out of 100000)
## done 38000 (out of 100000)
## done 40000 (out of 100000)
## done 42000 (out of 100000)
## done 44000 (out of 100000)
## done 46000 (out of 100000)
## done 48000 (out of 100000)
## done 50000 (out of 100000)
## done 52000 (out of 100000)
## done 54000 (out of 100000)
## done 56000 (out of 100000)
## done 58000 (out of 100000)
## done 60000 (out of 100000)
## done 62000 (out of 100000)
## done 64000 (out of 100000)
## done 66000 (out of 100000)
## done 68000 (out of 100000)
## done 70000 (out of 100000)
## done 72000 (out of 100000)
## done 74000 (out of 100000)
## done 76000 (out of 100000)
## done 78000 (out of 100000)
## done 80000 (out of 100000)
## done 82000 (out of 100000)
## done 84000 (out of 100000)
## done 86000 (out of 100000)
## done 88000 (out of 100000)
## done 90000 (out of 100000)
## done 92000 (out of 100000)
## done 94000 (out of 100000)
## done 96000 (out of 100000)
## done 98000 (out of 100000)
## time: 176s
## check counts
## trcnt,tecnt: 5000,5000
```

(d) Create a 2x2 table of the predicted test labels and true test labels. What does this tell you about the aforementioned concern?

```
library(gt)

# create a data frame so I can see the 0 and 1 labels - round to make the probabilities binary
raw_table <- cbind(c(0,1),
      table(round(binary.mod$prob.test.mean)),
      table(testdf$new))


# create a table with labels
as.tibble(raw_table) %>%
  mutate(Labels = V1,
         Preds = V2,
         True = V3) %>%
```

```
  select(Labels, Preds, True) %>%
  gt::gt()
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
```

Labels	Preds	True
0	96	75
1	2	23

Concern: "given that roughly five out of six of our observations take on the lowest fraud score, these fit statistics may simply be capturing the models' accurate prediction of cases with no obvious fraudulent activities (496).''

This table validates the aforementioned concern. There are fewer predicted fraud cases than actual fraud cases, meaning that the model's error is entirely on one side of the model and solely comes from mis-classifying fraudulent cases as not fraudulent. Because the training data had much more non-fraud cases, the model is better at predicting those.

Question 4

Which particular variables are most important in allowing the fitted informed forensic model to identify electoral fraud? We will answer this question using the following approach from the paper.

"When the number of trees is low, BART tends to rely on the most relevant predictors when building trees, as predictors are forced to compete to improve the model's fit (Chipman, George, and McCulloch 2010). As a result, we can re-estimate the BART model using a small number of trees (viz. 10) and calculate the average share of splitting rules that involve each variable (497).''

(a) Fit the informed forensics model (total 31 predictors) using the entire data set.

```
## (a)
set.seed(seed.num)
# TODO: specify the arguments (x.train, y.train) - the most recent x.train.fm and x.test.fm are include
bart.full.mod <- wbart(x.train = x.train.fm,
                       y.train = y.train,
                       k = 2, power = 0.5, base = 0.95,
                       sigdf = 10, sigquant = 0.75, nskip = 50000,
                       ndpost = 5000, ntree = 100, sigest = 0.35,
                       keepevery = 10, printevery = 2000)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 500, 31, 0
## y1,yn: -0.120037, 0.639549
## x1,x[n*p]: 0.076870, 0.000000
## *****Number of Trees: 100
## *****Number of Cut Points: 100 ... 1
## *****burn and ndpost: 50000, 50000
## *****Prior:beta,alpha,tau,nu,lambda: 0.500000,0.950000,0.062275,10.000000,0.082531
```

```
## *****sigma: 0.350000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,31,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 5000,5000,5000,5000
## *****printevery: 2000
## *****skiptr,skipte,skipteme,skiptreedraws: 10,10,10,10
##
## MCMC
## done 0 (out of 100000)
## done 2000 (out of 100000)
## done 4000 (out of 100000)
## done 6000 (out of 100000)
## done 8000 (out of 100000)
## done 10000 (out of 100000)
## done 12000 (out of 100000)
## done 14000 (out of 100000)
## done 16000 (out of 100000)
## done 18000 (out of 100000)
## done 20000 (out of 100000)
## done 22000 (out of 100000)
## done 24000 (out of 100000)
## done 26000 (out of 100000)
## done 28000 (out of 100000)
## done 30000 (out of 100000)
## done 32000 (out of 100000)
## done 34000 (out of 100000)
## done 36000 (out of 100000)
## done 38000 (out of 100000)
## done 40000 (out of 100000)
## done 42000 (out of 100000)
## done 44000 (out of 100000)
## done 46000 (out of 100000)
## done 48000 (out of 100000)
## done 50000 (out of 100000)
## done 52000 (out of 100000)
## done 54000 (out of 100000)
## done 56000 (out of 100000)
## done 58000 (out of 100000)
## done 60000 (out of 100000)
## done 62000 (out of 100000)
## done 64000 (out of 100000)
## done 66000 (out of 100000)
## done 68000 (out of 100000)
## done 70000 (out of 100000)
## done 72000 (out of 100000)
## done 74000 (out of 100000)
## done 76000 (out of 100000)
## done 78000 (out of 100000)
## done 80000 (out of 100000)
## done 82000 (out of 100000)
## done 84000 (out of 100000)
## done 86000 (out of 100000)
## done 88000 (out of 100000)
## done 90000 (out of 100000)
```

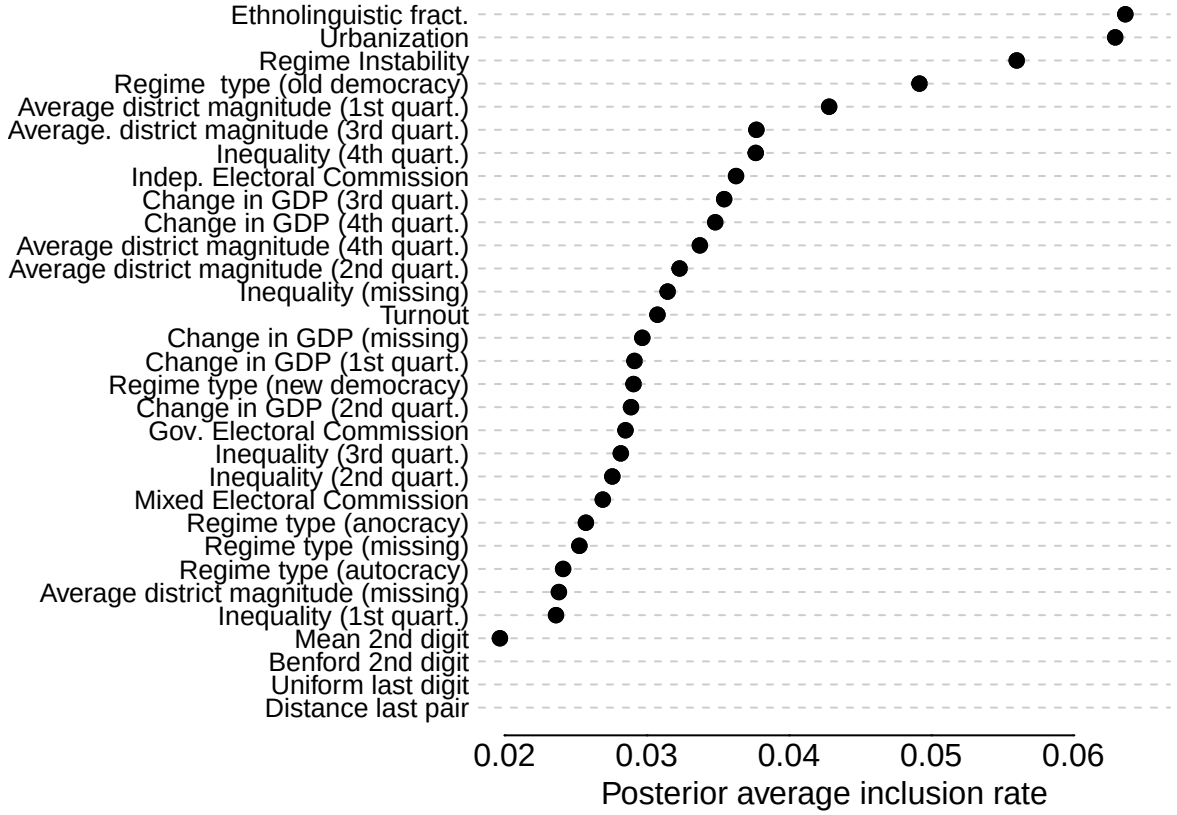
```
## done 92000 (out of 100000)
## done 94000 (out of 100000)
## done 96000 (out of 100000)
## done 98000 (out of 100000)
## time: 182s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 5000,0,0,5000
```

(b) Replicate Figure 2 using the item `varcount` from the fitted output. Briefly discuss the results.

Hint:

- `varcount`: a matrix with ndpost rows and nrow(x.train) columns. Each row is for a draw. For each variable (corresponding to the columns), the total count of the number of times that variable is used in a tree decision rule (over all trees) is given.
- Sample codes are as below.

```
## (b)
## Get nr. of variables (a.k.a. features) used
numVars <- length(colMeans(bart.full.mod$varcount))

## Calculate relative frequency of feature usage
## in splitting rules, and sort
values <- colMeans(prop.table(bart.full.mod$varcount,1))
ordering <- order(values)

## More legible names to variables (TODO: you may need to change the order)
names <- c("Uniform last digit", "Distance last pair", "Mean 2nd digit",
           "Benford 2nd digit", "Ethnolinguistic fract.",  "Urbanization",
           "Turnout", "Regime type (anocracy)", "Regime type (autocracy)",
           "Regime type (missing)", "Regime type (new democracy)",
           "Regime  type (old democracy)", "Inequality (1st quart.)",
           "Inequality (2nd quart.)", "Inequality (3rd quart.)", "Inequality (4th quart.)",
           "Inequality (missing)", "Average district magnitude (1st quart.)",
           "Average district magnitude (2nd quart.)", "Average. district magnitude (3rd quart.)",
           "Average district magnitude (4th quart.)", "Average district magnitude (missing)",
           "Change in GDP (1st quart.)", "Change in GDP (2nd quart.)", "Change in GDP (3rd quart.)",
           "Change in GDP (4th quart.)", "Change in GDP (missing)", "Gov. Electoral Commission",
           "Mixed Electoral Commission", "Indep. Electoral Commission", "Regime Instability")

## Create Graph
par(yaxt="n", mfrow=c(1,1), mar=c(2.5,14,.5,.5), bty="n", mgp=c(1,0,0), tcl=0)
plot(values[ordering], 1:numVars, ylab="", pch=19, main=""
     , xlab="Posterior average inclusion rate"
     ,xlim=c(0.02,0.065))
par(las=1)
mtext(names[ordering], at=1:numVars, cex=.8, side=2, line=.25)
for(i in 1:numVars){
  abline(h=i, lty=2, col="gray80")
}
points(values[ordering], 1:numVars, ylab="", pch=19)
```



The above graph shows the posterior average inclusion rate for each of the variables. The posterior average inclusion rate shows the proportion of steps in the MCMC chain that include the variable. Ethnolinguistic fictionalization has the greatest rate, meaning that it is the most likely to be included in the real model - meaning it likely is the best predictor of election fraud. Regime instability and regime type were the second and third respectively variables with the highest rates.