Feature Selection Using Random Forest In Multicollinearity Settings

rong.lu@utsw.edu

10/10/2019

Logistic Regression Simulation Design 1:

This simulation is designed to show variable importance rankings generated by random forest are not ranking variables according to their effect size in multicollinearity settings under the multivariate logistic regression.

In the following simulation, there are 3 types of model inputs/features: x's, c's, and f's. The 20 x's are independent sample from standard normal distribution. The 20 c's are generated from multivariate normal distribution with all mean=0, var=1, and pairwise correlation=0.8. Both x's and c's are used in simulation of response y under a logistic regression model with all slope coefficient =1.

The 20 f's are fake model inputs. We generate f's using the same method as generating x's, but do not use them to simulate response variable y.

Since all f's are not used to simulate response y, the effect size of all f's should be 0.

Since all x's are indepedent and have equal mean, variance, and regression coefficients, the effect size of all x's should be the same.

Since all c's have the same mean, variance, regression coefficients, and all pairwise correlation are the same, we'd expect the effect size of all c's are the same.

Since all x's and c's have the same mean, variance, regression coefficients, and only c's are correlated, we'd expect the effect size of every c is bigger than the effect size of every x.

Logistic Regression Simulation Example 1

```
library(MASS)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1234)
# generate starting values of the inputs
n = 2000                  # sample size in the simulation dataset
p = 20
mu = rep(0,p)

x1 = mvrnorm(n, mu, diag(1,p,p))      # the independent inputs
colnames(x1) <- paste("x",1:p,sep="")

sigma <- matrix(0.8,p,p)
diag(sigma) <- 1
x2 = mvrnorm(n, mu, sigma)            # correlated inputs
colnames(x2) <- paste("c",1:p,sep="")
```

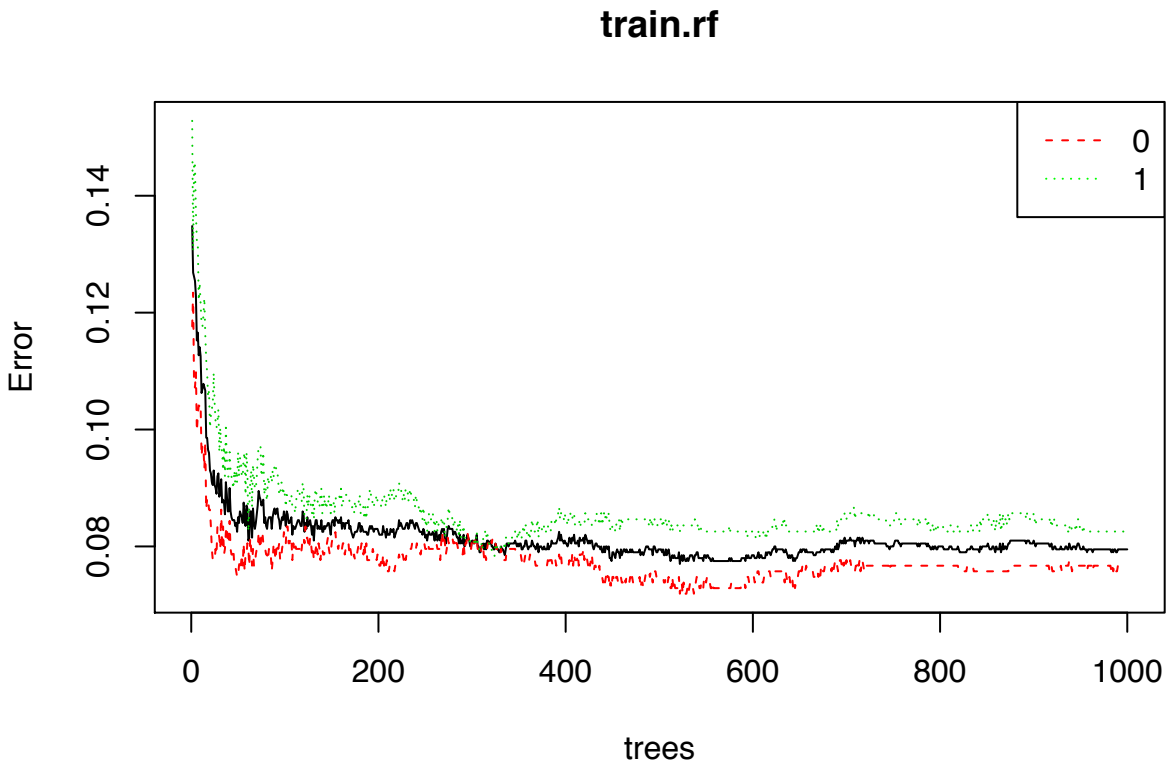
```r
inputs <- cbind(x1, x2)

# simulate the response variable y from logistic regression:
bias <- 0.1
coeff <- rep(1, p*2)
z = bias + coeff%*%t(inputs)        # linear combination with a bias
pr = 1/(1+exp(-z))                  # pass through an inv-logit function
y = rbinom(n,1,pr)                  # bernoulli response variable

# check balance:
table(y)
```

```
## y
##    0    1
## 1043  957
```

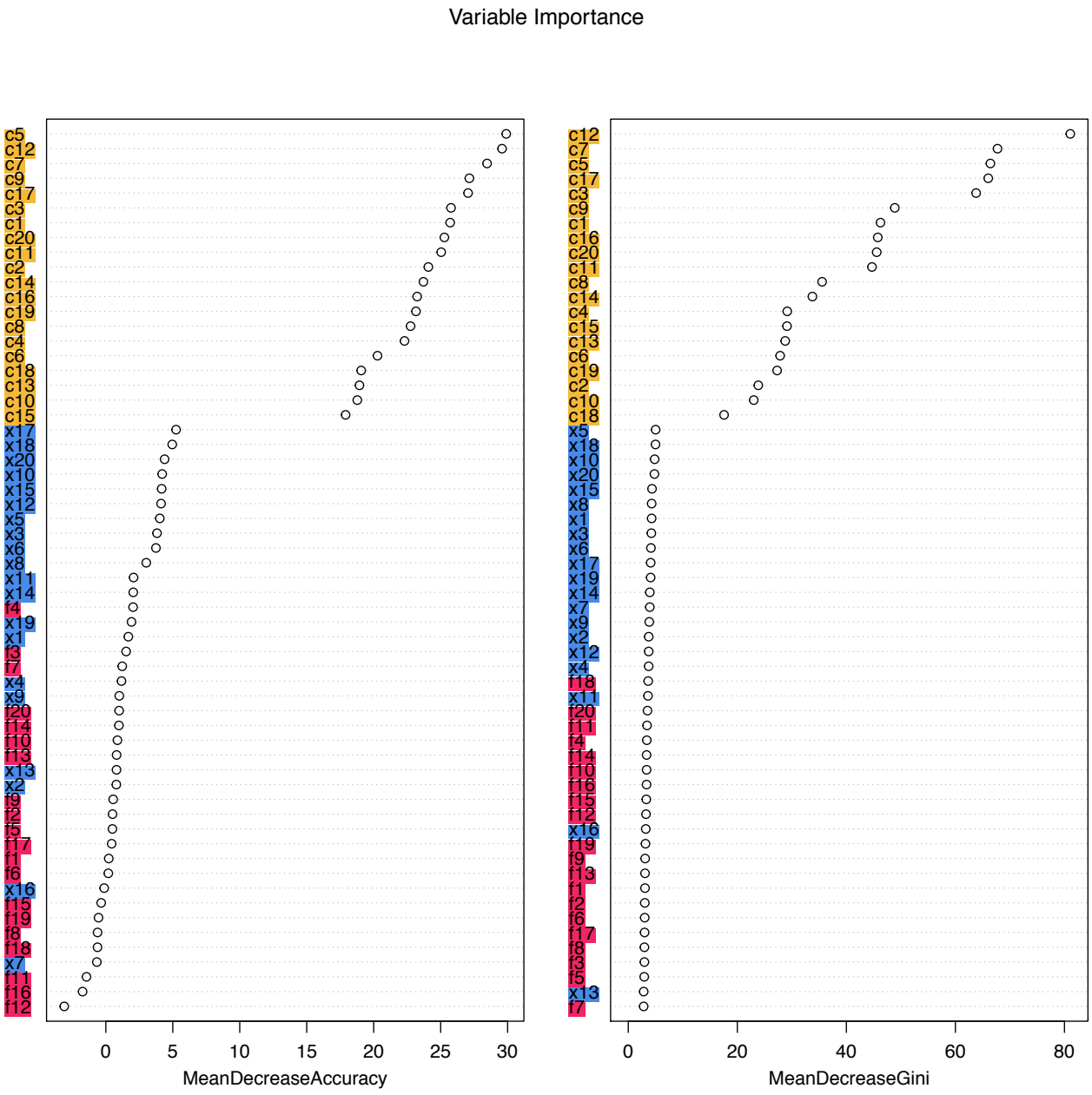
```r
# now generate some fake inputs and fit random forest to generate importance rankings:
fake.x = mvrnorm(n, mu, diag(1,p,p))      # the independent fake inputs
colnames(fake.x) <- paste("f",1:p,sep="")

train <- cbind(x1,x2,fake.x)
train.rf <- randomForest(train, factor(y),
                         ntree=1000,importance=T, keep.forest = T)
train.rf$err.rate[1000,]
```

```
##        OOB          0          1
## 0.07950000 0.07670182 0.08254963
```

```r
plot(train.rf);legend("topright", legend=levels(factor(y)), lty=2:3, col=c("red","green"));
```



```
varImpPlot(train.rf, sort=T, main="Variable Importance", n.var = 60);
```



Logistic Regression Simulation Design 2:

This simulation is designed to show variable importance rankings generated by random forest are not ranking variables according to their effect size in multicollinearity settings under the multivariate logistic regression.

In the following simulation, there are 4 types of model inputs/features: 10 x's, 10 c's, 10 sc's and 20 f's. The 10 x's are independent sample from standard normal distribution. The 10 c's are generated from multivariate normal distribution with all mean=0, var=1, and pairwise correlation=0.3. The 10 sc's are generated from multivariate normal distribution with all mean=0, var=1, and pairwise correlation=0.8. All x's, c's and sc's are used in simulation of response y under a logistic regression model with all slope coefficient =1.

The 30 f's are fake model inputs. We generate f's using the same method as generating x's, but do not use them to simulate response variable y.

Since all f's are not used to simulate response y, the effect size of all f's should be 0. Since all x's are indepedent and have equal mean, variance, and regression coefficients, the effect size of all x's should be the same.

Since all c's have the same mean, variance, regression coefficients, and all pairwise correlation are the same, we'd expect the effect size of all c's are the same.

Since all x's, c's and sc's have the same mean, variance, regression coefficients, we'd expect the effect size of every c is bigger than the effect size of every x, and the effect size of every sc is bigger than the effect size of every c.

Logistic Regression Simulation Example 2

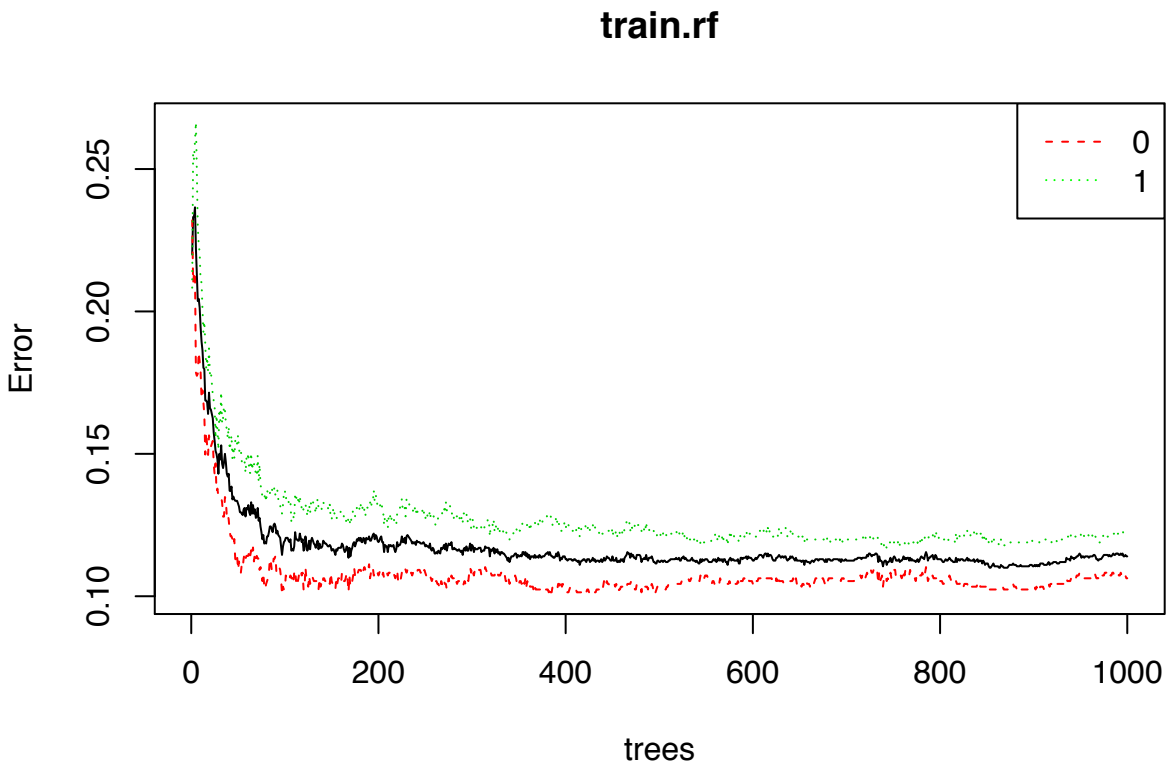
```
library(MASS)
library(randomForest)

set.seed(1234)
# generate starting values of the inputs
n = 2000                  # sample size in the simulation dataset
p = 10
mu = rep(0,p)

x1 = mvrnorm(n, mu, diag(1,p,p))       # the independent inputs
colnames(x1) <- paste("x",1:p,sep="")

sigma <- matrix(0.3,p,p)
diag(sigma) <- 1
x2 = mvrnorm(n, mu, sigma)             # correlated inputs
colnames(x2) <- paste("c",1:p,sep="")

sigma <- matrix(0.8,p,p)
diag(sigma) <- 1
x3 = mvrnorm(n, mu, sigma)             # correlated inputs
colnames(x3) <- paste("sc",1:p,sep="")

inputs <- cbind(x1, x2, x3)

# simulate the response variable y from logistic regression:
bias <- 0.1
coeff <- rep(1, p*3)
z = bias + coeff%*%t(inputs)         # linear combination with a bias
pr = 1/(1+exp(-z))                   # pass through an inv-logit function
y = rbinom(n,1,pr)                   # bernoulli response variable

# check balance:
table(y)
```

```
## y
##    0    1
## 1016  984
```

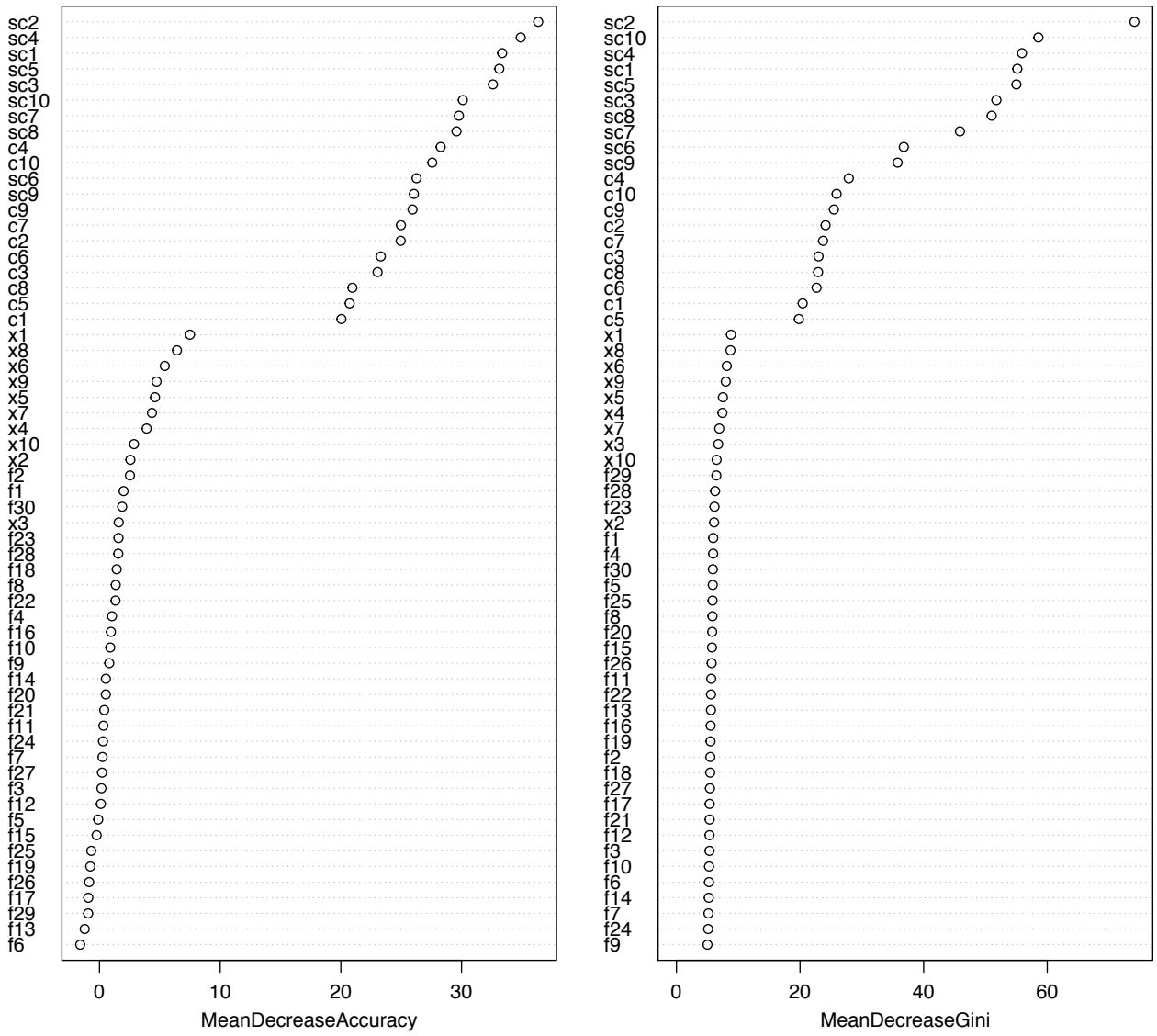
```
# now generate some fake inputs and fit random forest to generate importance rankings:
fake.x = mvrnorm(n, rep(0,30), diag(1,30,30))      # the independent fake inputs
colnames(fake.x) <- paste("f",1:30,sep="")

train <- cbind(inputs,fake.x)
train.rf <- randomForest(train, factor(y),
                         ntree=1000,importance=T, keep.forest = T)
train.rf$err.rate[1000,]
```

```
##       OOB         0         1
## 0.1140000 0.1062992 0.1219512
```

```
plot(train.rf);legend("topright", legend=levels(factor(y)), lty=2:3, col=c("red","green"));
```




```
varImpPlot(train.rf, sort=T, main="Variable Importance", n.var = 60);
```


Variable Importance
sc2
sc4
sc1
sc5
sc3
sc10
sc7
sc8
c4
c10
sc6
sc9
c9
c7
c2
c6
c3
c8
c5
c1
x1
x8
x6
x9
x5
x7
x4
x10
x2
f2
f1
f30
x3
f23
f28
f18
f8
f22
f4
f16
f10
f9
f14
f20
f21
f11
f24
f7
f27
f3
f12
f5
f15
f25
f19
f26
f17
f29
f13
f6
0
10
20
30
MeanDecreaseAccuracy
sc2
sc10
sc4
sc1
sc5
sc3
sc8
sc7
sc6
sc9
c4
c10
c9
c2
c7
c3
c8
c6
c1
c5
x1
x8
x6
x9
x5
x4
x7
x3
x10
f29
f28
f23
x2
f1
f4
f30
f5
f25
f8
f20
f15
f26
f11
f22
f13
f16
f19
f2
f18
f27
f17
f21
f12
f3
f10
f6
f14
f7
f24
f9
0
20
40
60
MeanDecreaseGini