Classification, Tree, and Random Forest

(Jing Li, Miami University)

1. This note introduces classification that predicts a binary dependent variable. Examples are predicting whether an email is a spam, whether a viewer is interested in a video, and whether a person votes for Trump.

2. We use wage data[1], and the focus is on the binary variable **married**, which equals one for a married person, and zero for an unmarried person. The predictors or features include wage (wa), education (ed), female (fe), and others.

```
> da = read.csv("wagedata.csv")
> head(da)
  wage educ exper female married numdep tenure
1  3.1   11     2      1       0      2      0
2  3.2   12    22      1       1      3      2
3  3.0   11     2      0       0      2      0
4  6.0    8    44      0       1      0     28
5  5.3   12     7      0       1      1      2
6  8.8   16     9      0       1      0      8
> names(da) = c("wa","ed","ex","fe","ma","nu","te")
> str(da)
'data.frame':        526 obs. of  7 variables:
 $ wa: num  3.1 3.2 3 6 5.3 8.8 11 5 3.6 18 ...
 $ ed: int  11 12 11 8 12 16 18 12 12 17 ...
 $ ex: int  2 22 2 44 7 9 15 5 26 22 ...
 $ fe: int  1 1 0 0 0 0 0 1 1 0 ...
 $ ma: int  0 1 0 1 1 1 0 0 0 1 ...
 $ nu: int  2 3 2 0 1 0 0 0 2 0 ...
 $ te: int  0 2 0 28 2 8 7 3 4 21 ...
```

   There are 526 observations, and seven variables. We rename each variable using the first two letters.

3. An important fact about dummy variable is that its mean value is the same as proportion of that variable being equal to one. That is, let D be a dummy variable, we

[1] *https* : *//fsb.miamioh.edu/lij*14/411_*wage.xls*

have

$$E(D) = P(D = 1) \tag{1}$$

To see this,

```
> table(da$ma)
  0   1
206 320
> prop.table(table(da$ma))
       0        1
0.391635 0.608365
> mean(da$ma)
[1] 0.608365
```

In wage data 320 persons are married, and 206 are unmarried. The proportion of married persons is 0.608365, the same as the mean value 0.608365. Later we will just use the R mean function to compute proportion for dummy variables.

4. We first consider the predictor female (fe), which is a dummy variable itself—equaling one for female, and zero for male.

```
> table(da$fe)
  0   1
274 252
> table(da$fe,da$ma)
     0   1
  0  86 188
  1 120 132
> prop.table(table(da$fe,da$ma),margin=1)
           0         1
  0 0.3138686 0.6861314
  1 0.4761905 0.5238095
```

There are 274 males, and 252 females. Among the 274 males, 86 are unmarried, and 188 are married. The proportion of married males is 0.6861314. The proportion of married females is 0.5238095. The two proportions seem different, implying that gender may matter for likelihood of being married.

5. The 2 by 2 table reported by `table(da$fe,da$ma)` is called two-way table. Based on the two-way table, we can conduct a chi-squared test for the null hypothesis that married and female are independent:

```
> chisq.test(da$fe,da$ma,correct=F)
        Pearson's Chi-squared test
data:  da$fe and da$ma
X-squared = 14.517, df = 1, p-value = 0.0001389
```

The p-value 0.0001389 is less than 0.05, so the null hypothesis is rejected. That can be seen as the formal evidence that gender matters for marital status since the two variables are not independent. As a result, gender should be considered as a predictor.

6. The relationship between female and married can be quantified by a linear probability model (LPM):

```
> summary(lm(da$ma~da$fe))$coef
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  0.6861314 0.02913383 23.551019 3.433689e-84
da$fe       -0.1623219 0.04209109 -3.856442 1.293623e-04
```

The intercept 0.6861314 is $E(married|female = 0)$, the proportion of married males. The slope -0.1623219 is $E(married|female = 1) - E(married|female = 0) = 0.5238095 - 0.6861314$, the difference between the proportion of married females and the proportion of married males. The t value -3.856442 implies that the difference is statistically significant (gender matters!), consistent with the chi-squared test. Note that the proportion of married females can be inferred from the LPM as $0.6861314 - 0.1623219 = 0.5238095$.

Tree Model with Binary Predictor

7. Alternatively, we can consider a decision tree, which splits the sample (root) into two subsamples (branches). First, we need to install two packages.

```
> install.packages("rpart")
> install.packages("rpart.plot")
> library(rpart)
> library(rpart.plot)
```

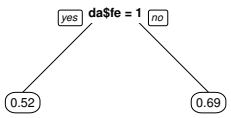
The tree is built by the rpart function

```
> tree1=rpart(da$ma~da$fe,method="anova",control=rpart.control(maxdepth=1))
> tree1
n= 526
node), split, n, deviance, yval
      * denotes terminal node

1) root 526 125.32320 0.6083650
  2) da$fe>=0.5 252  62.85714 0.5238095 *
  3) da$fe< 0.5 274  59.00730 0.6861314 *
```

```
> prp(tree1)
```

where the prp function draws tree1 as



(a) At node 1 (root, whole sample), there are 526 observations. The unconditional deviance (total sum square) of married is 125.32320, and unconditional mean of married is 0.6083650, see codes below

```
> length(da$ma)
[1] 526
> sum((da$ma-mean(da$ma))^2)
[1] 125.3232
> mean(da$ma)
[1] 0.608365
```

(b) At node 2 (left branch, sub-sample 1), when female is greater than or equal to 0.5 (or female equals one), there are 252 observations. The conditional deviance is 62.85714, and conditional mean is 0.5238095, see codes below

```
> length(da$ma[da$fe==1])
[1] 252
> sum((da$ma[da$fe==1]-mean(da$ma[da$fe==1]))^2)
[1] 62.85714
> mean(da$ma[da$fe==1])
[1] 0.5238095
```

(c) Codes below explain the numbers at node 3 (right branch, sub-sample 2)

```
> length(da$ma[da$fe==0])
[1] 274
> sum((da$ma[da$fe==0]-mean(da$ma[da$fe==0]))^2)
[1] 59.0073
> mean(da$ma[da$fe==0])
[1] 0.6861314
```

(d) Now we can predict probability of being married for each person in the sample. The first six predicted probabilities are

```
> head(predict(tree1,da))
        1         2         3         4         5         6
0.5238095 0.5238095 0.6861314 0.6861314 0.6861314 0.6861314
> head(da$fe)
[1] 1 1 0 0 0 0
```

Basically, depending on whether female is 1 (greater than or equal to 0.5) or 0 (less than 0.5), the tree model uses the conditional mean 0.5238095 or 0.6861314, which is also conditional proportion, as predicted probabilities. Then we can classify married as one if the probability is greater than 0.5

```
> yhat_tree1 = ifelse(predict(tree1,da)>0.5,1,0)
> head(yhat_tree1)
1 1 1 1 1 1
> head(da$ma)
[1] 0 1 0 1 1 1
```

Compared to the actual value of married, we see that the first and third persons are mis-classified.

(e) It is clear that this way of classification is not informative at all—since both conditional means 0.5238095 and 0.6861314 are greater than 0.5, we always predict that a person be married. This undesirable situation suggests that we may compare 0.5238095 and 0.6861314 to the unconditional mean 0.6083650 other than 0.5

```
> yhat_tree1_new = ifelse(predict(tree1,da)>mean(da$ma),1,0)
> head(yhat_tree1_new)
0 0 1 1 1 1
> head(da$ma)
[1] 0 1 0 1 1 1
```

Now the classification varies across persons, but second and third persons are mis-classified

8. Notice that the key results of the tree model, the two conditional means, can be obtained from LPM as well. Thus, for a binary predictor (or categorical feature with only two levels), LPM and tree model deliver the same results. There is no advantage of trying the tree model if the predictor is dummy variable.

Categorical Predictor with More Than Two Levels

9. The benefit of tree model can be noticeable for a categorical predictor with more than two levels. For instance, the variable nu has seven levels.

```
> table(da$nu)
  0   1   2   3   4   5   6
252 105  99  45  16   7   2
```

The results of LPM that predicts married with nu are

```
> summary(lm(da$ma~factor(da$nu)))$coef
                   Estimate Std. Error    t value     Pr(>|t|)
(Intercept)     0.51190476 0.03017793 16.9628873 1.135401e-51
factor(da$nu)1  0.14523810 0.05564535  2.6100672 9.313929e-03
factor(da$nu)2  0.21536797 0.05682316  3.7901444 1.682265e-04
factor(da$nu)3  0.28809524 0.07752850  3.7159916 2.243858e-04
factor(da$nu)4  0.05059524 0.12350849  0.4096499 6.822318e-01
factor(da$nu)5 -0.08333333 0.18356516 -0.4539714 6.500392e-01
factor(da$nu)6  0.48809524 0.34008796  1.4352029 1.518316e-01
```

(a) The rule is that we should use a categorical predictor as factor. The reference level is 0 (for which the dummy variable is excluded to avoid dummy variable trap), and the intercept 0.51190476 is the conditional mean of married for that level

```
> mean(da$ma[da$nu==0])
[1] 0.5119048
```

(b) Other coefficients are differences between that level and the reference level. For instance, the coefficient of the dummy variable for level 1 (labeled as `factor(da$nu)1`) is

```
> mean(da$ma[da$nu==1])-mean(da$ma[da$nu==0])
[1] 0.1452381
```

(c) Since there are only seven and two observations for levels 5 and 6, the LPM may suffer overfitting issue by capturing noise other than signal.

(d) The t values of 0.4096499 and -0.4539714 for levels 4 and 5 indicate that there are no differences between those two levels and level 0

10. Amazingly, the tree model is able to take into account the fact that levels of 0, 4, and 5 might be considered as just one category (since they are not different in terms of effect on married):

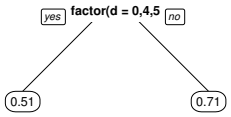
```
> tree1nb=rpart(da$ma~factor(da$nu),method="anova")
> tree1nb

1) root 526 125.32320 0.6083650
  2) factor(da$nu)=0,4,5 275  68.70545 0.5127273 *
  3) factor(da$nu)=1,2,3,6 251  51.34661 0.7131474 *
```

Codes below explain the numbers on node 2

```
> length(da$ma[da$nu%in%c(0,4,5)])
[1] 275
> sum((da$ma[da$nu%in%c(0,4,5)]-mean(da$ma[da$nu%in%c(0,4,5)]))^2)
[1] 68.70545
> mean(da$ma[da$nu%in%c(0,4,5)])
[1] 0.5127273
```

The tree looks like



(a) To predict married, we use the left branch when nu equals 0, 4, 5. The conditional mean is 0.5127273.

(b) We use the right branch when nu equals 1, 2, 3, 6. The conditional mean is 0.7131474

(c) This tree model does not suffer the over-fitting issue since levels 5 and 6 (noises) are not used to create extra individual branches.

(d) We can compare this tree to tree1, which uses female as the predictor

```
> tree1
1) root 526 125.32320 0.6083650
  2) da$fe>=0.5 252  62.85714 0.5238095 *
  3) da$fe< 0.5 274  59.00730 0.6861314 *

> as.logical(68.70545+51.34661<62.85714+59.00730)
[1] TRUE
```

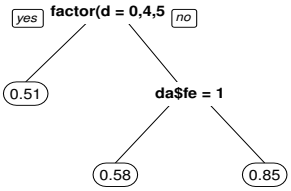
Recall that deviance measures prediction error. Based on deviance, the left branch using nu $= 0, 4, 5$ produces worse classification than the left branch using fe=1 (because 68.70545>62.85714); whereas the right branch using nu=1,2,3,6 produces better classification than the right branch using fe=0 (because 51.34661<59.00730). Overall, nu has greater predictive power than female (because $68.70545+51.34661 < 62.85714 + 59.00730$)

11. Next we include both female and nu in the tree model

```
> tree1nc=rpart(da$ma~factor(da$nu)+da$fe,method="anova")
> tree1nc

1) root 526 125.32320 0.6083650
  2) factor(da$nu)=0,4,5 275  68.70545 0.5127273 *
  3) factor(da$nu)=1,2,3,6 251  51.34661 0.7131474
    6) da$fe>=0.5 130  31.56923 0.5846154 *
    7) da$fe< 0.5 121  15.32231 0.8512397 *
```

The tree looks like



(Exercise): describe how to predict married using this tree.

Tree Model with Continuous Predictor

12. The benefit of tree model can be even more remarkable when (i) the predictor is continuous and when (ii) nonlinearity or interaction is present.

13. For instance, the variable wage (wa) is numeric and continuous. The LPM that uses wa without and with its squared term are

```
> summary(lm(da$ma~da$wa))$coef
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 0.43097866 0.039079601 11.028226 1.417692e-25
da$wa       0.03001973 0.005602991  5.357805 1.263821e-07
> da$wasq = da$wa^2
> summary(lm(da$ma~da$wa+da$wasq))$coef
                Estimate   Std. Error   t value     Pr(>|t|)
(Intercept)  0.260430529 0.0691113088  3.768277 1.830830e-04
da$wa        0.080451514 0.0178050023  4.518478 7.707304e-06
da$wasq     -0.002619927 0.0008786916 -2.981623 3.000848e-03
```

The negative coefficient -0.002619927 of the squared term implies a parabola facing downward—as wage rises, the chance of being married first rises, and beyond a

turning point, starts to fall. Its t value -2.981623 implies that (i) the squared term is statistically significant and (ii) nonlinearity is present.

14. The regression model is parametric in the sense that it imposes a specific function form, a quadratic function in this case, for the conditional mean. By contrast, the tree model is non-parametric without assuming a specific function form. In fact, the tree model is based on step function, which can approximate any function in a short window (bandwidth). For example, consider the simplest tree that uses wa as predictor

```
> tree2=rpart(da$ma~da$wa,method="anova",control=rpart.control(maxdepth=1))
> tree2

1) root 526 125.32320 0.6083650
  2) da$wa< 3.75 176  43.03977 0.4261364 *
  3) da$wa>=3.75 350  73.50000 0.7000000 *
```

The key message from tree2 is that people with wage greater than 3.75 has higher chance of being married (conditional mean is 0.7000000) than people with wage less than 3.75 (conditional mean is 0.4261364)

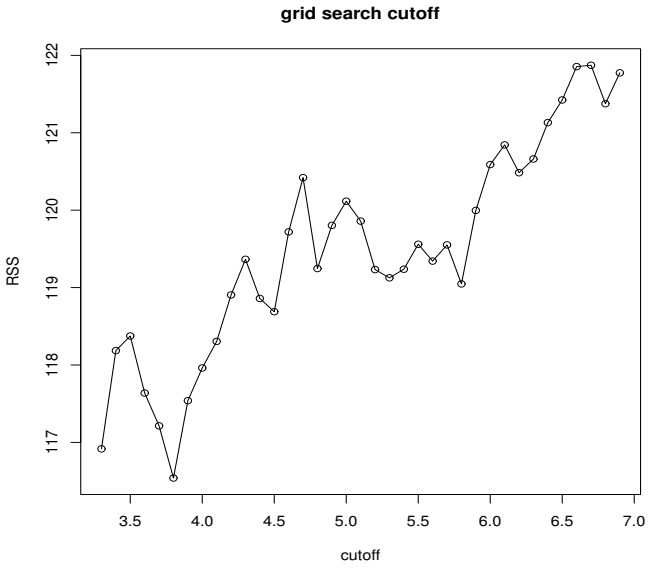
15. The data-driven cutoff value 3.75 is a crucial intermediate output that is obtained through a grid search that minimizes the residual sum squares

```
> summary(da$wa)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.530   3.300   4.700   5.909  6.900  25.000
> ir = seq(3.3,6.9,0.1)
> v.cut = rep(NA,length(ir))
> v.rss = rep(NA,length(ir))
> j = 1
> for (i in ir) {
+  v.cut[j] = i
+  x = factor(da$wa>=i)
+  uhat = resid(lm(da$ma~x))
+  v.rss[j] = sum(uhat^2)
+  j = j+1
+ }
```

```
> ocut = v.cut[which(v.rss==min(v.rss))]
> cat("optimal cut is",ocut,"\n")
optimal cut is 3.8

> length(da$ma[da$wa<ocut])
[1] 176
> sum((da$ma[da$wa<ocut]-mean(da$ma[da$wa<ocut]))^2)
[1] 43.03977
> mean(da$ma[da$wa<ocut])
[1] 0.4261364
```

Basically, we treat each value between the first and third quartiles of wage as a potential cutoff. Then we use a R for loop to find the optimal cutoff that produces the least deviance (residual sum squares). Finally, we use the optimal cutoff to split the sample into two sub-samples, and create the tree. The grid search plot is below



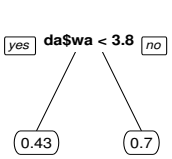
16. Of course we let the tree grow by setting $maxdepth = 2$

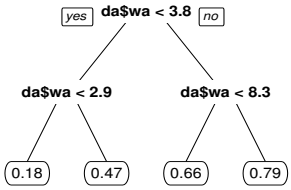
```
> tree3=rpart(da$ma~da$wa,method="anova",control=rpart.control(maxdepth=2))
> tree3

1) root 526 125.323200 0.6083650
  2) da$wa< 3.75 176  43.039770 0.4261364
```

```
    4) da$wa< 2.85 28   4.107143 0.1785714 *
    5) da$wa>=2.85 148  36.891890 0.4729730 *
  3) da$wa>=3.75 350  73.500000 0.7000000
    6) da$wa< 8.25 248  55.548390 0.6612903 *
    7) da$wa>=8.25 102  16.676470 0.7941176 *
> mean(da$ma)
[1] 0.608365
```

However, tree3 turns out to be no more useful than tree2. Notice that the conditional means at node 4 and 5 are both less than the unconditional mean 0.608365, while the conditional means at node 6 and 7 are both greater than 0.608365. This finding indicates that classification from tree3 is the same as tree2. Or in other words, tree3 is unnecessarily complex, so needs pruning. The tree2 and tree3 look like





Confusion Matrix

17. An important tool to evaluate the performance of classification models is confusion matrix, a table that displays the counts of true positive, true negative, false positive, and false negative predictions. For instance, the confusion matrix for LPM without the squared term of wage is

```
> # confusion matrix
> yhat_reg = ifelse(fitted(lm(da$ma~da$wa))>mean(da$ma),1,0)
> table(da$ma,yhat_reg)
   yhat_reg
      0   1
  0 153  53
  1 176 144
```

The meaning of each number is indicated by following codes

```
> length(which(da$ma==0&yhat_reg==0))
[1] 153
> length(which(da$ma==1&yhat_reg==0))
[1] 176
> length(which(da$ma==0&yhat_reg==1))
[1] 53
> length(which(da$ma==1&yhat_reg==1))
[1] 144
```

We see that the model mis-classifies $176 + 53 = 229$ persons. After the squared term is added, the prediction improves

```
> yhat_sqreg = ifelse(fitted(lm(da$ma~da$wa+da$wasq))>mean(da$ma),1,0)
> table(da$ma,yhat_sqreg)
   yhat_sqreg
      0   1
  0 145  61
  1 153 167
```

Now, the number of mis-classifications becomes $153+61 = 214$. Confusion matrix below show that tree2 and tree3 have the same number of mis-classifications 180. Thus tree3 needs pruning, and tree2 is the best classification model (by having the least sum of terms not on the diagonal line in the confusion matrix).

```
> yhat_tree2 = ifelse(predict(tree2)>mean(da$ma),1,0)
> table(da$ma,yhat_tree2)
   yhat_tree2
```

```
     0   1
  0 101 105
  1  75 245
>
> yhat_tree3 = ifelse(predict(tree3)>mean(da$ma),1,0)
> table(da$ma,yhat_tree3)
   yhat_tree3
     0   1
  0 101 105
  1  75 245
```

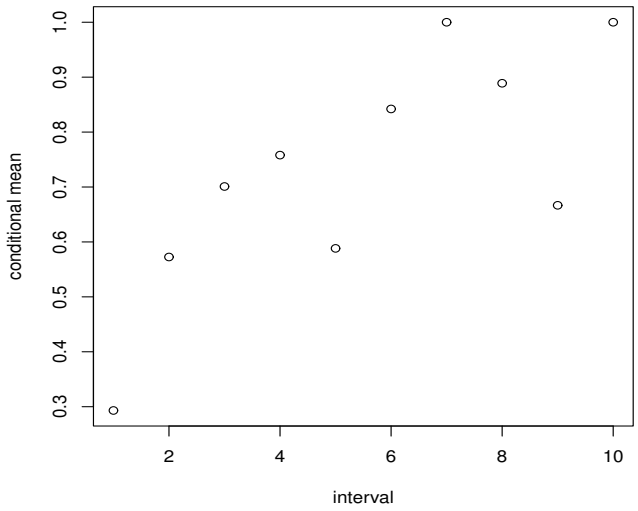
To sum up, the off-diagonal values of a confusion matrix are crucial for understanding the errors made by the classification model. When ranking classifications, we look for the model that produces the least sum of off-diagonal values.

Nonlinearity

18. The main reason why tree2 outperforms LPM with the squared term is that in this case the nonlinearity does not take the form of a quadratic function. To see this, we can divide wage into ten intervals, and compute the conditional mean of married for each interval

```
> intervals = cut(da$wa, breaks = 10)
> tapply(da$ma, intervals, mean)
(0.506,2.98]  (2.98,5.42]  (5.42,7.87]  (7.87,10.3]  (10.3,12.8]  (12.8,15.2]  (15.
   0.2931034    0.5725806    0.7009346    0.7580645    0.5882353    0.8421053    1.
 (17.7,20.1]  (20.1,22.6]    (22.6,25]
   0.8888889    0.6666667    1.0000000
```

The graph below plots the conditional mean in each interval, and it is clear that the pattern is highly nonlinear, and does not look like a parabola.



Tree vs Regression

19. A tree model may outperform a regression model in several scenarios:

    (a) If the relationship between predictors and the dependent variable is highly nonlinear, a decision tree may outperform a regression model, which typically assumes a specific function form such as linear and quadratic.

    (b) Decision trees can handle interactions between predictors more effectively than a regression model.

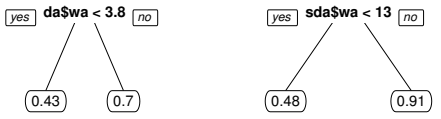
    (c) If the data contains outliers that significantly affect the regression model's performance, a decision tree may provide better results.

    (d) If the data contains a mix of numerical and categorical predictors, a decision tree may outperform a regression model.

Random Forest

20. Tree model has the limitation that it does not provide standard error, which measures the sampling uncertainty. Moreover, the tree model can suffer big variance—after the sample changes, the results from tree model can vary a lot. For instance, the graph below compares tree2 to tree2n that only uses the first 100 observations of wage data.

```
tree2=rpart(da$ma~da$wa,method="anova",control=rpart.control(maxdepth=1))
sda = da[1:100,]
tree2n=rpart(sda$ma~sda$wa,method="anova",control=rpart.control(maxdepth=1))
par(mfrow=c(1,2))
prp(tree2)
prp(tree2n)
```



We see the cutoff changes from 3.8 to 13. The conditional means change accordingly.

21. Random Forest aims to improve the tree model in at least two ways. First, it accounts for the sampling uncertainty by (i) resampling data repeatedly (called bootstrap), (ii) creating multiple trees (forest) based on those resampled data; and (iii) averaging the predicted probabilities obtained from the multiple trees.

22. Bootstrap entails resampling the index of data with replacement. For instance, consider the first five observations. One resampled data look like

```
> da[1:5,]
   wa ed ex fe ma nu te
1 3.1 11  2  1  0  2  0
2 3.2 12 22  1  1  3  2
3 3.0 11  2  0  0  2  0
4 6.0  8 44  0  1  0 28
5 5.3 12  7  0  1  1  2
> index = sample(1:5,5,replace=T)
```

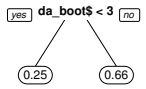
```
> index
[1] 1 4 4 5 2
> da[index,]
     wa ed ex fe ma nu te
1   3.1 11  2  1  0  2  0
4   6.0  8 44  0  1  0 28
4.1 6.0  8 44  0  1  0 28
5   5.3 12  7  0  1  1  2
2   3.2 12 22  1  1  3  2
```

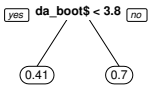
In this case, the 4th observations appear twice in the bootstrap data, whereas the third observation is absent. Loosely speaking, resampling data is like reshuffling a deck of cards. It enables us to re-use the original data repeatedly. By doing so, we can account for the sampling variability.

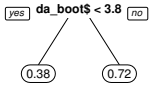
23. There is a technical issue. We want to ensure that the bootstrap data preserve the unconditional mean of married. That is, we hope to maintain the ratio of married vs unmarried persons. Thus, conventional bootstrap does not work here, and we need stratified bootstrap that resamples data within married and unmarried persons. Codes below illustrates the stratified bootstrap

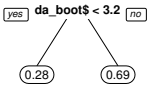
```
> # stratified bootstrap
> da_one = da[da$ma==1,]
> n_one = nrow(da_one)
> da_zero = da[da$ma==0,]
> n_zero = nrow(da_zero)
>
> inde_one = sample(seq(1,n_one),n_one,replace=T)
> inde_zero = sample(seq(1,n_zero),n_zero,replace=T)
> da_boot = rbind(da_one[inde_one,],da_zero[inde_zero,])
> mean(da_boot$ma)
[1] 0.608365
> mean(da$ma)
[1] 0.608365
```

24. An an illustration, next we bootstrap data four times, and create four trees (a mini forest)









(Exercise): can you explain why the second and third trees have different conditional means despite that they have the same cutoff values.

25. For instance, suppose a person has wage 4. The average predicted probability of being married for this person is

$$\frac{0.6580087 + 0.7042254 + 0.7223796 + 0.6863208}{4} = 0.6580087 \tag{2}$$

where we get the numbers in the numerator from the appropriate branch in each individual tree (this person is on the right branch for all four trees). Because 0.6580087 exceeds the unconditional mean 0.608365, this person is classified as being married. The codes to obtain that mini forest is below

```
> forest = list()
> for (i in 1:4) {
+     inde_one = sample(seq(1,n_one),n_one,replace=T)
+     inde_zero = sample(seq(1,n_zero),n_zero,replace=T)
+     da_boot = rbind(da_one[inde_one,],da_zero[inde_zero,])
+     tree2b=rpart(da_boot$ma~da_boot$wa,method="anova",control=rpart.control(maxde
+     forest[[i]]=tree2b
+ }
```

The details of the first tree are below

```
> forest[[1]]
n= 526

1) root 526 125.3232 0.6083650
  2) da_boot$wa< 2.95 64  12.0000 0.2500000 *
  3) da_boot$wa>=2.95 462 103.9654 0.6580087 *
```

The codes to obtain the predict probability of being married from each tree are

```
> for (i in 1:4) {
+ cat("predicted probability of being married from tree ", i, "is","\n")
+ print(predict(forest[[i]],newdata)[nrow(da)])
+ }
predicted probability of being married from tree  1 is
      526
0.6580087
predicted probability of being married from tree  2 is
      526
0.7042254
predicted probability of being married from tree  3 is
      526
0.7223796
predicted probability of being married from tree  4 is
      526
0.6863208

> mean(0.6580087,0.7042254,0.7223796,0.6863208)
[1] 0.6580087
```

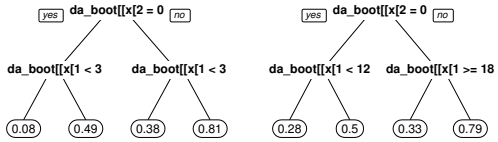
26. The second way by which random forest improves the tree model is that a random subset of predictors is used to create individual tree. For instance, tree1 may use wa and te as predictors; tree2 may use ed and te, see codes below. By choosing a random set of predictors, random forest is able to (i) mitigate the issue of collinearity and outlier, and (ii) reduce the variance of average forecast.

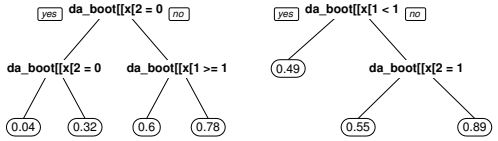
```
> p.set = c("wa","ed","ex","fe","nu","te")
> set.seed(1234)
> forest = list()
> for (i in 1:4) {
+     inde_one = sample(seq(1,n_one),n_one,replace=T)
+     inde_zero = sample(seq(1,n_zero),n_zero,replace=T)
+     da_boot = rbind(da_one[inde_one,],da_zero[inde_zero,])
+     x = sample(p.set,2,replace=F)
+     cat("*****************","","\n")
+     cat("predictors in tree",i, "are ",x,"\n")
+     tree2b=rpart(da_boot$ma~da_boot[[x[1]]]+da_boot[[x[2]]],method="anova",contro
+     forest[[i]]=tree2b
+ }
*****************
predictors in tree 1 are  wa te
*****************
predictors in tree 2 are  ed te
*****************
predictors in tree 3 are  fe ex
*****************
predictors in tree 4 are  nu fe
```

Pay attention to these two lines in particular

```
x = sample(p.set,2,replace=F)
rpart(da_boot$ma~da_boot[[x[1]]]+da_boot[[x[2]]],method="anova",control=rpart.contr
```

The new random forest with randomly chosen predictors looks like





27. In practice, we just use the built-in randomForest function. For instance, below we use the last observation (the 526th) of wage data as testing set, and all other observations as training set. The random forest consists of 100 tree. The 526th observation is classified as 1.

```
> install.packages("randomForest")
> library(randomForest)
> set.seed(12345)
> fit = randomForest(factor(ma)~wa+ed+fe+nu, da[-526,],ntree=100)
> predict(fit, da[526,])
526
  1
Levels: 0 1
# to learn more about this function
> ? randomForest
```

28. Random forest can also be applied to numeric dependent variable. Then it is called regression tree as opposed to decision tree. For instance, we can build a random forest to predict wage of a married male with 10 year education. Below are codes and results

```
> set.seed(12345)
> fit = randomForest(wa~ed+fe+ma, da, ntree=100)
> predict(fit, data.frame(ed=10,fe=0,ma=1))
```

```
       1
6.198521
```

The predicted wage is 6.198521. This prediction makes sense—it is less than 8, the average wage of married male, since his education is below 12.89894, the average education in that group .

```
> mean(da$wa[da$ma==1&da$fe==0])
[1] 8
> mean(da$ed[da$ma==1&da$fe==0])
[1] 12.89894
```

29. We can compare the sum of squared forecasting error (SSFE) between OLS regression and random forest. First, we use ed, ma and fe to predict wa. We consider out-of-sample forecast by excluding the unit we want to predict (in a fashion called rolling window)

```
> set.seed(12345)
> yhat_ols = rep(NA,nrow(da))
> yhat_rf = rep(NA,nrow(da))
> for (i in 1:nrow(da)) {
+  m_ols = lm(wa~ed+fe+ma,da[-i,])
+  yhat_ols[i]=predict(m_ols,da[i,])
+  m_rf = randomForest(wa~ed+fe+ma,da[-i,],ntree=100)
+  yhat_rf[i]=predict(m_rf,da[i,])
+ }
> sum((da$wa-yhat_ols)^2)
[1] 5267.823
> sum((da$wa-yhat_rf)^2)
[1] 5229.609
```

We see random forest outperforms because its SSFE 5229.609 is less than that of OLS regression 5267.823. Next we add ex and nu as predictors

```
> set.seed(12345)
> yhat_ols = rep(NA,nrow(da))
```

```
> yhat_rf = rep(NA,nrow(da))
> for (i in 1:nrow(da)) {
+  m_ols = lm(wa~ed+fe+ma+nu+ex,da[-i,])
+  yhat_ols[i]=predict(m_ols,da[i,])
+  m_rf = randomForest(wa~ed+fe+ma+nu+ex,da[-i,],ntree=100)
+     yhat_rf[i]=predict(m_rf,da[i,])
+ }
> sum((da$wa-yhat_ols)^2)
[1] 5035.794
> sum((da$wa-yhat_rf)^2)
[1] 4819.696
```

Now the superiority of random forest becomes more noticeable.

30. There may be multiple reasons why random forest performs better in this case. One reason is that the OLS regression above fails to account for nonlinearity and interaction. For instance, the new OLS regression below indicates that squared term of ex and interaction term of fe and ma should be considered.

```
> summary(lm(wa~ed+fe+ma+nu+ex+I(ex^2)+ma:fe,da))$coef
                Estimate   Std. Error    t value     Pr(>|t|)
(Intercept) -3.028419978 0.7869217089 -3.8484387 1.337176e-04
ed           0.541728908 0.0526964669 10.2801751 1.101491e-22
fe          -0.460346306 0.4191129321 -1.0983825 2.725478e-01
ma           1.603832228 0.4195183507  3.8230324 1.478565e-04
nu          -0.072210971 0.1135418483 -0.6359855 5.250668e-01
ex           0.246270825 0.0381624698  6.4532203 2.518490e-10
I(ex^2)     -0.004360541 0.0008369092 -5.2102923 2.726578e-07
fe:ma       -2.687261585 0.5377102104 -4.9976019 7.953261e-07
```

Nevertheless, even this new regression may ignore other possible nonlinearity and interaction. Also, this new regression has the drawback that it treats nu as numeric other than categorical. By contrast, the random forest automatically takes into account of mix of numeric and categorical variables, and possible nonlinearity and interaction.

31. In general, the benefit of random forest grows as the number of predictors rises (issues such as mixed type, nonlinearity, interaction, and collinearity rise).