

Classification and Regression Trees 1

author: Dr. Marko Mitic

Problem Description: **PREDICTING EARNINGS FROM CENSUS DATA**

The United States government periodically collects demographic information by conducting a census.

In this problem, we are going to use census information about an individual to predict how much a person earns – in particular, whether the person earns more than \$50,000 per year. This data comes from the UCI Machine Learning Repository <http://archive.ics.uci.edu/ml/datasets/Adult>.

The file census.csv contains 1994 census data for 31,978 individuals in the United States.

The dataset includes the following 13 variables:

- **age** = the age of the individual in years
- **workclass** = the classification of the individual's working status (does the person work for the federal government, work for the local government, work without pay, and so on)
- **education** = the level of education of the individual (e.g., 5th-6th grade, high school graduate, PhD, so on)
- **maritalstatus** = the marital status of the individual
- **occupation** = the type of work the individual does (e.g., administrative/clerical work, farming/fishing, sales and so on)
- **relationship** = relationship of individual to his/her household
- **race** = the individual's race
- **sex** = the individual's sex
- **capitalgain** = the capital gains of the individual in 1994 (from selling an asset such as a stock or bond for more than the original purchase price)
- **capitalloss** = the capital losses of the individual in 1994 (from selling an asset such as a stock or bond for less than the original purchase price)
- **hoursperweek** = the number of hours the individual works per week
- **nativecountry** = the native country of the individual
- **over50k** = whether or not the individual earned more than \$50,000 in 1994

Firstly, let's begin by building a logistic regression model to predict whether an individual's earnings are above \$50,000 (the variable "over50k") using all of the other variables as independent variables. Read the dataset census.csv into R.

Next, let's split the data randomly into a training set and a testing set, setting the seed to 2000 before creating the split. The data is splitted in such a way that the training set contains 60% of the observations, while the testing set contains 40% of the observations.

We want to build a logistic regression model to predict the dependent variable "over50k", using all of the other variables in the dataset as independent variables. We'll use the training set to build the model.

```
census = read.csv("census.csv")
str(census)
```

```
## 'data.frame': 31978 obs. of 13 variables:
## $ age : int 39 50 38 53 28 37 49 52 31 42 ...
## $ workclass : Factor w/ 9 levels " ?"," Federal-gov",...: 8 7 5 5 5 5 5 7 5 5 ...
## $ education : Factor w/ 16 levels " 10th"," 11th",...: 10 10 12 2 10 13 7 12 13 10 ...
## $ maritalstatus: Factor w/ 7 levels " Divorced"," Married-AF-spouse",...: 5 3 1 3 3 3 4 3 5 3 ...
## $ occupation : Factor w/ 15 levels " ?"," Adm-clerical",...: 2 5 7 7 11 5 9 5 11 5 ...
## $ relationship : Factor w/ 6 levels " Husband"," Not-in-family",...: 2 1 2 1 6 6 2 1 2 1 ...
## $ race : Factor w/ 5 levels " Amer-Indian-Eskimo",...: 5 5 5 3 3 5 3 5 5 5 ...
## $ sex : Factor w/ 2 levels " Female"," Male": 2 2 2 2 1 1 1 2 1 2 ...
## $ capitalgain : int 2174 0 0 0 0 0 0 0 14084 5178 ...
## $ capitalloss : int 0 0 0 0 0 0 0 0 0 0 ...
## $ hoursperweek : int 40 13 40 40 40 40 16 45 50 40 ...
## $ nativecountry: Factor w/ 41 levels " Cambodia"," Canada",...: 39 39 39 39 5 39 23 39 39 39 ...
## $ over50k : Factor w/ 2 levels " <=50K"," >50K": 1 1 1 1 1 1 1 2 2 2 ...
```

```
#install.packages("caTools")
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.2.1
```

```
set.seed(2000)
split = sample.split(census$over50k, SplitRatio = 0.6)
train = subset(census, split == T)
test = subset(census, split == F)

LogReg = glm(over50k~., data = train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(LogReg)
```

```
##
## Call:
## glm(formula = over50k ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -5.1065  -0.5037  -0.1804  -0.0008   3.3383
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error z value
## (Intercept)   -8.658e+00  1.379e+00 -6.279
## age           2.548e-02  2.139e-03 11.916
## workclass Federal-gov  1.105e+00  2.014e-01  5.489
## workclass Local-gov   3.675e-01  1.821e-01  2.018
## workclass Never-worked -1.283e+01  8.453e+02 -0.015
## workclass Private     6.012e-01  1.626e-01  3.698
## workclass Self-emp-inc  7.575e-01  1.950e-01  3.884
```

## workclass Self-emp-not-inc	1.855e-01	1.774e-01	1.046
## workclass State-gov	4.012e-01	1.961e-01	2.046
## workclass Without-pay	-1.395e+01	6.597e+02	-0.021
## education 11th	2.225e-01	2.867e-01	0.776
## education 12th	6.380e-01	3.597e-01	1.774
## education 1st-4th	-7.075e-01	7.760e-01	-0.912
## education 5th-6th	-3.170e-01	4.880e-01	-0.650
## education 7th-8th	-3.498e-01	3.126e-01	-1.119
## education 9th	-1.258e-01	3.539e-01	-0.355
## education Assoc-acdm	1.602e+00	2.427e-01	6.601
## education Assoc-voc	1.541e+00	2.368e-01	6.506
## education Bachelors	2.177e+00	2.218e-01	9.817
## education Doctorate	2.761e+00	2.893e-01	9.544
## education HS-grad	1.006e+00	2.169e-01	4.638
## education Masters	2.421e+00	2.353e-01	10.289
## education Preschool	-2.237e+01	6.864e+02	-0.033
## education Prof-school	2.938e+00	2.753e-01	10.672
## education Some-college	1.365e+00	2.195e-01	6.219
## maritalstatus Married-AF-spouse	2.540e+00	7.145e-01	3.555
## maritalstatus Married-civ-spouse	2.458e+00	3.573e-01	6.880
## maritalstatus Married-spouse-absent	-9.486e-02	3.204e-01	-0.296
## maritalstatus Never-married	-4.515e-01	1.139e-01	-3.962
## maritalstatus Separated	3.609e-02	1.984e-01	0.182
## maritalstatus Widowed	1.858e-01	1.962e-01	0.947
## occupation Adm-clerical	9.470e-02	1.288e-01	0.735
## occupation Armed-Forces	-1.008e+00	1.487e+00	-0.677
## occupation Craft-repair	2.174e-01	1.109e-01	1.960
## occupation Exec-managerial	9.400e-01	1.138e-01	8.257
## occupation Farming-fishing	-1.068e+00	1.908e-01	-5.599
## occupation Handlers-cleaners	-6.237e-01	1.946e-01	-3.204
## occupation Machine-op-inspct	-1.862e-01	1.376e-01	-1.353
## occupation Other-service	-8.183e-01	1.641e-01	-4.987
## occupation Priv-house-serv	-1.297e+01	2.267e+02	-0.057
## occupation Prof-specialty	6.331e-01	1.222e-01	5.180
## occupation Protective-serv	6.267e-01	1.710e-01	3.664
## occupation Sales	3.276e-01	1.175e-01	2.789
## occupation Tech-support	6.173e-01	1.533e-01	4.028
## occupation Transport-moving	NA	NA	NA
## relationship Not-in-family	7.881e-01	3.530e-01	2.233
## relationship Other-relative	-2.194e-01	3.137e-01	-0.699
## relationship Own-child	-7.489e-01	3.507e-01	-2.136
## relationship Unmarried	7.041e-01	3.720e-01	1.893
## relationship Wife	1.324e+00	1.331e-01	9.942
## race Asian-Pac-Islander	4.830e-01	3.548e-01	1.361
## race Black	3.644e-01	2.882e-01	1.265
## race Other	2.204e-01	4.513e-01	0.488
## race White	4.108e-01	2.737e-01	1.501
## sex Male	7.729e-01	1.024e-01	7.545
## capitalgain	3.280e-04	1.372e-05	23.904
## capitalloss	6.445e-04	4.854e-05	13.277
## hoursperweek	2.897e-02	2.101e-03	13.791
## nativecountry Canada	2.593e-01	1.308e+00	0.198
## nativecountry China	-9.695e-01	1.327e+00	-0.730
## nativecountry Columbia	-1.954e+00	1.526e+00	-1.280

```

## nativecountry Cuba 5.735e-02 1.323e+00 0.043
## nativecountry Dominican-Republic -1.435e+01 3.092e+02 -0.046
## nativecountry Ecuador -3.550e-02 1.477e+00 -0.024
## nativecountry El-Salvador -6.095e-01 1.395e+00 -0.437
## nativecountry England -6.707e-02 1.327e+00 -0.051
## nativecountry France 5.301e-01 1.419e+00 0.374
## nativecountry Germany 5.474e-02 1.306e+00 0.042
## nativecountry Greece -2.646e+00 1.714e+00 -1.544
## nativecountry Guatemala -1.293e+01 3.345e+02 -0.039
## nativecountry Haiti -9.221e-01 1.615e+00 -0.571
## nativecountry Holand-Netherlands -1.282e+01 2.400e+03 -0.005
## nativecountry Honduras -9.584e-01 3.412e+00 -0.281
## nativecountry Hong -2.362e-01 1.492e+00 -0.158
## nativecountry Hungary 1.412e-01 1.555e+00 0.091
## nativecountry India -8.218e-01 1.314e+00 -0.625
## nativecountry Iran -3.299e-02 1.366e+00 -0.024
## nativecountry Ireland 1.579e-01 1.473e+00 0.107
## nativecountry Italy 6.100e-01 1.333e+00 0.458
## nativecountry Jamaica -2.279e-01 1.387e+00 -0.164
## nativecountry Japan 5.072e-01 1.375e+00 0.369
## nativecountry Laos -6.831e-01 1.661e+00 -0.411
## nativecountry Mexico -9.182e-01 1.303e+00 -0.705
## nativecountry Nicaragua -1.987e-01 1.507e+00 -0.132
## nativecountry Outlying-US(Guam-USVI-etc) -1.373e+01 8.502e+02 -0.016
## nativecountry Peru -9.660e-01 1.678e+00 -0.576
## nativecountry Philippines 4.393e-02 1.281e+00 0.034
## nativecountry Poland 2.410e-01 1.383e+00 0.174
## nativecountry Portugal 7.276e-01 1.477e+00 0.493
## nativecountry Puerto-Rico -5.769e-01 1.357e+00 -0.425
## nativecountry Scotland -1.188e+00 1.719e+00 -0.691
## nativecountry South -8.183e-01 1.341e+00 -0.610
## nativecountry Taiwan -2.590e-01 1.350e+00 -0.192
## nativecountry Thailand -1.693e+00 1.737e+00 -0.975
## nativecountry Trinidad&Tobago -1.346e+00 1.721e+00 -0.782
## nativecountry United-States -8.594e-02 1.269e+00 -0.068
## nativecountry Vietnam -1.008e+00 1.523e+00 -0.662
## nativecountry Yugoslavia 1.402e+00 1.648e+00 0.851
## Pr(>|z|)
## (Intercept) 3.41e-10 ***
## age < 2e-16 ***
## workclass Federal-gov 4.03e-08 ***
## workclass Local-gov 0.043641 *
## workclass Never-worked 0.987885
## workclass Private 0.000218 ***
## workclass Self-emp-inc 0.000103 ***
## workclass Self-emp-not-inc 0.295646
## workclass State-gov 0.040728 *
## workclass Without-pay 0.983134
## education 11th 0.437738
## education 12th 0.076064 .
## education 1st-4th 0.361897
## education 5th-6th 0.516008
## education 7th-8th 0.263152
## education 9th 0.722228

```

## education Assoc-acdm	4.10e-11 ***
## education Assoc-voc	7.74e-11 ***
## education Bachelors	< 2e-16 ***
## education Doctorate	< 2e-16 ***
## education HS-grad	3.52e-06 ***
## education Masters	< 2e-16 ***
## education Preschool	0.973996
## education Prof-school	< 2e-16 ***
## education Some-college	5.00e-10 ***
## maritalstatus Married-AF-spouse	0.000378 ***
## maritalstatus Married-civ-spouse	6.00e-12 ***
## maritalstatus Married-spouse-absent	0.767155
## maritalstatus Never-married	7.42e-05 ***
## maritalstatus Separated	0.855672
## maritalstatus Widowed	0.343449
## occupation Adm-clerical	0.462064
## occupation Armed-Forces	0.498170
## occupation Craft-repair	0.049972 *
## occupation Exec-managerial	< 2e-16 ***
## occupation Farming-fishing	2.15e-08 ***
## occupation Handlers-cleaners	0.001353 **
## occupation Machine-op-inspct	0.176061
## occupation Other-service	6.14e-07 ***
## occupation Priv-house-serv	0.954385
## occupation Prof-specialty	2.22e-07 ***
## occupation Protective-serv	0.000248 ***
## occupation Sales	0.005282 **
## occupation Tech-support	5.63e-05 ***
## occupation Transport-moving	NA
## relationship Not-in-family	0.025562 *
## relationship Other-relative	0.484263
## relationship Own-child	0.032716 *
## relationship Unmarried	0.058392 .
## relationship Wife	< 2e-16 ***
## race Asian-Pac-Islander	0.173504
## race Black	0.206001
## race Other	0.625263
## race White	0.133356
## sex Male	4.52e-14 ***
## capitalgain	< 2e-16 ***
## capitalloss	< 2e-16 ***
## hoursperweek	< 2e-16 ***
## nativecountry Canada	0.842879
## nativecountry China	0.465157
## nativecountry Columbia	0.200470
## nativecountry Cuba	0.965432
## nativecountry Dominican-Republic	0.962972
## nativecountry Ecuador	0.980829
## nativecountry El-Salvador	0.662181
## nativecountry England	0.959686
## nativecountry France	0.708642
## nativecountry Germany	0.966572
## nativecountry Greece	0.122527
## nativecountry Guatemala	0.969180

```

## nativecountry Haiti 0.568105
## nativecountry Holand-Netherlands 0.995736
## nativecountry Honduras 0.778775
## nativecountry Hong 0.874155
## nativecountry Hungary 0.927653
## nativecountry India 0.531661
## nativecountry Iran 0.980736
## nativecountry Ireland 0.914628
## nativecountry Italy 0.647194
## nativecountry Jamaica 0.869467
## nativecountry Japan 0.712179
## nativecountry Laos 0.680866
## nativecountry Mexico 0.481103
## nativecountry Nicaragua 0.895132
## nativecountry Outlying-US(Guam-USVI-etc) 0.987115
## nativecountry Peru 0.564797
## nativecountry Philippines 0.972640
## nativecountry Poland 0.861624
## nativecountry Portugal 0.622327
## nativecountry Puerto-Rico 0.670837
## nativecountry Scotland 0.489616
## nativecountry South 0.541809
## nativecountry Taiwan 0.847878
## nativecountry Thailand 0.329678
## nativecountry Trinidad&Tobago 0.434105
## nativecountry United-States 0.946020
## nativecountry Vietnam 0.507799
## nativecountry Yugoslavia 0.394874
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 21175 on 19186 degrees of freedom
## Residual deviance: 12104 on 19090 degrees of freedom
## AIC: 12298
##
## Number of Fisher Scoring iterations: 15

```

Firstly, we can observe the **warning** message that indicates that our model is overfitting due to large number of independent variables. By typing **summary** command, we find that practically every variable except of **race** and **nativecountry** is significant (we use p-value threshold of 0.1). The accuracy of our logistic regression model can be determined on the test set:

```
pred = predict(LogReg, newdata = test, type = "response")
```

```

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading

```

```
table(test$over50k, pred>0.5)
```

```
##
```

```
##          FALSE TRUE
##    <=50K   9051  662
##    >50K    1190 1888
```

```
ACU = (9051+1888)/nrow(test)
ACU
```

```
## [1] 0.8552107
```

This must be compared to the baseline model as follows:

```
table(test$over50k)
```

```
##
##    <=50K   >50K
##    9713    3078
```

```
ACU = 9713/nrow(test)
ACU
```

```
## [1] 0.7593621
```

The results confirm that our model is good and that is fairly better than baseline model. This can be further confirmed using ROCR and AUC:

```
#install.packages("ROCR")
library(ROCR)
```

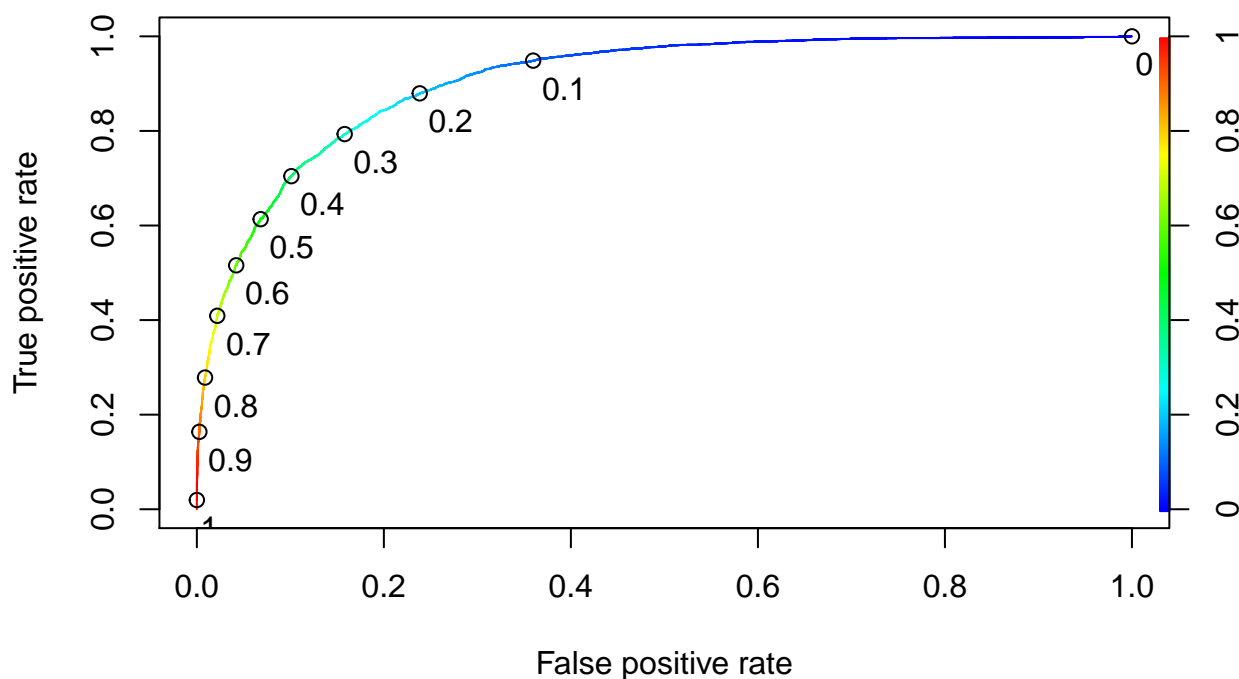
```
## Warning: package 'ROCR' was built under R version 3.2.1
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.2.1
```

```
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##    lowess
```

```
ROCPred2 = prediction(pred, test$over50k)
ROCRperf2=performance(ROCPred2, "tpr", "fpr")
plot(ROCRperf2, colorize=T, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))
```

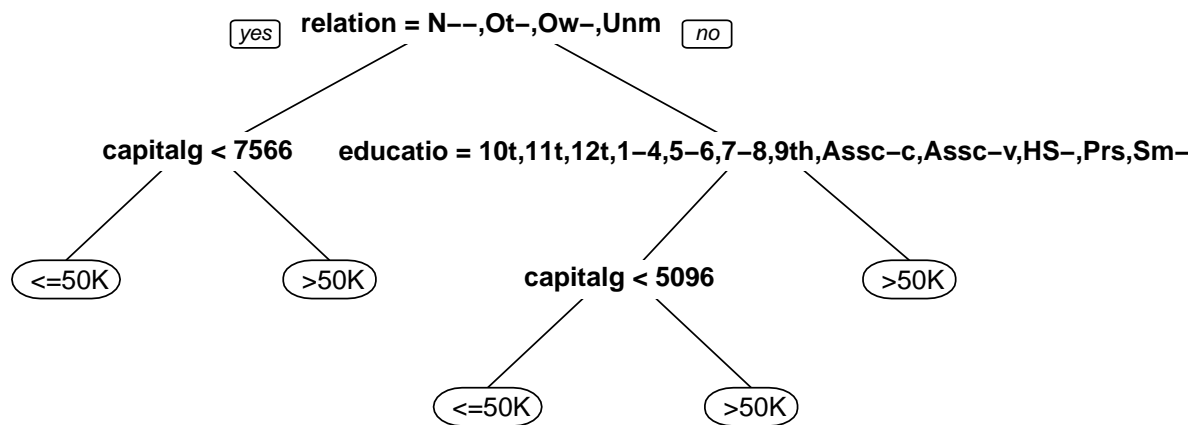


```
AUC = as.numeric(performance(ROCpred2, "auc")@y.values)
AUC
```

```
## [1] 0.9061598
```

Let us now build a classification tree to predict “over50k”. We’ll use the training set to build the model, and all of the other variables as independent variables.

```
#install.packages("rpart")
library(rpart)
#install.packages("rpart.prp")
library(rpart.plot)
CART = rpart(over50k~., data = train, method = "class")
prp(CART)
```

We can observe that the CART tree has 4 split, with relationship variable on top. Next, we're going to calculate accuracy of the CART model on test set:

```
pred = predict(CART, newdata = test, type="class")
```

```
table(test$over50k, pred)
```

```
##      pred
##      <=50K >50K
## <=50K   9243   470
## >50K   1482  1596
```

```
ACU =(9243+1596)/nrow(test)
ACU
```

```
## [1] 0.8473927
```

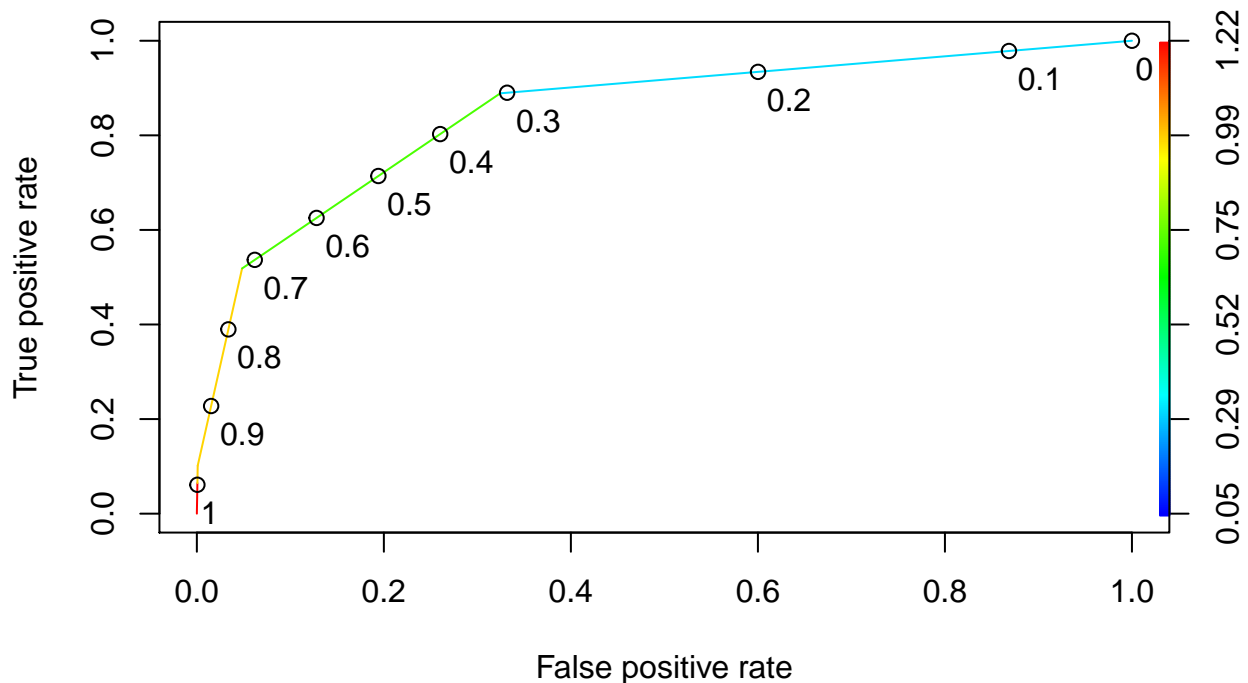
This highlights a very regular phenomenon when comparing CART and logistic regression. CART often performs a little worse than logistic regression in out-of-sample accuracy. However, as is the case here, the CART model is often much simpler to describe and understand.

Let us now consider the ROC curve and AUC for the CART model on the test set. They can be easily obtained:

```

pred2 = predict(CART, newdata = test)
ROCpred3 = prediction(pred2[,2], test$over50k)
ROCRperf3=performance(ROCpred3, "tpr", "fpr")
plot(ROCRperf3, colorize=T, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-0.2,1.7))

```



```

AUC = as.numeric(performance(ROCpred3, "auc")@y.values)
AUC

```

```
## [1] 0.8470256
```

Before building a random forest model, we'll down-sample our training set. While some modern personal computers can build a random forest model on the entire training set, others might run out of memory when trying to train the model since random forests is much more computationally intensive than CART or Logistic Regression. For this reason, before continuing we will define a new training set to be used when building our random forest model, that contains 2000 randomly selected observations from the original training set.

```

set.seed(1)
trainSmall = train[sample(nrow(train), 2000), ]

```

Let us now build a random forest model to predict “over50k”, using the dataset “trainSmall” as the data used to build the model.

```
#install.packages("randomForest")
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.1
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
rf = randomForest(over50k ~., data=trainSmall)
```

The accuracy of the model on the test set can be determined through:

```
pred2 = predict(rf, newdata=test)
table(test$over50k, pred2)
```

```
##      pred2
##      <=50K >50K
## <=50K   9586   127
## >50K   1985  1093
```

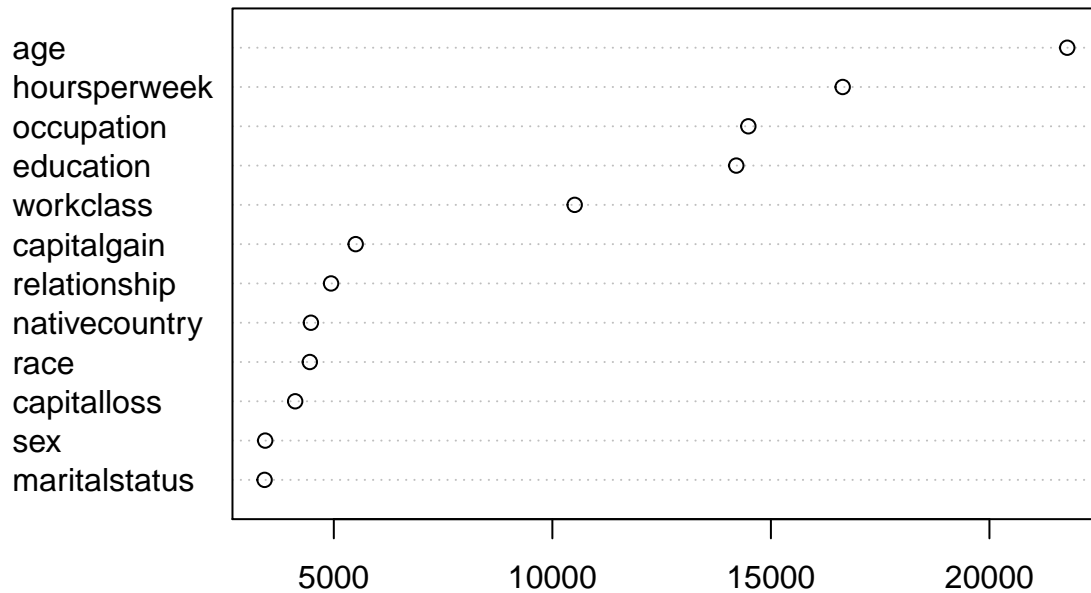
```
ACU = (9586+1093)/nrow(test)
ACU
```

```
## [1] 0.8348839
```

Random forest models work by building a large collection of trees. As a result, we lose some of the interpretability that comes with CART in terms of seeing how predictions are made and which variables are important. However, we can still compute metrics that give us insight into which variables are important.

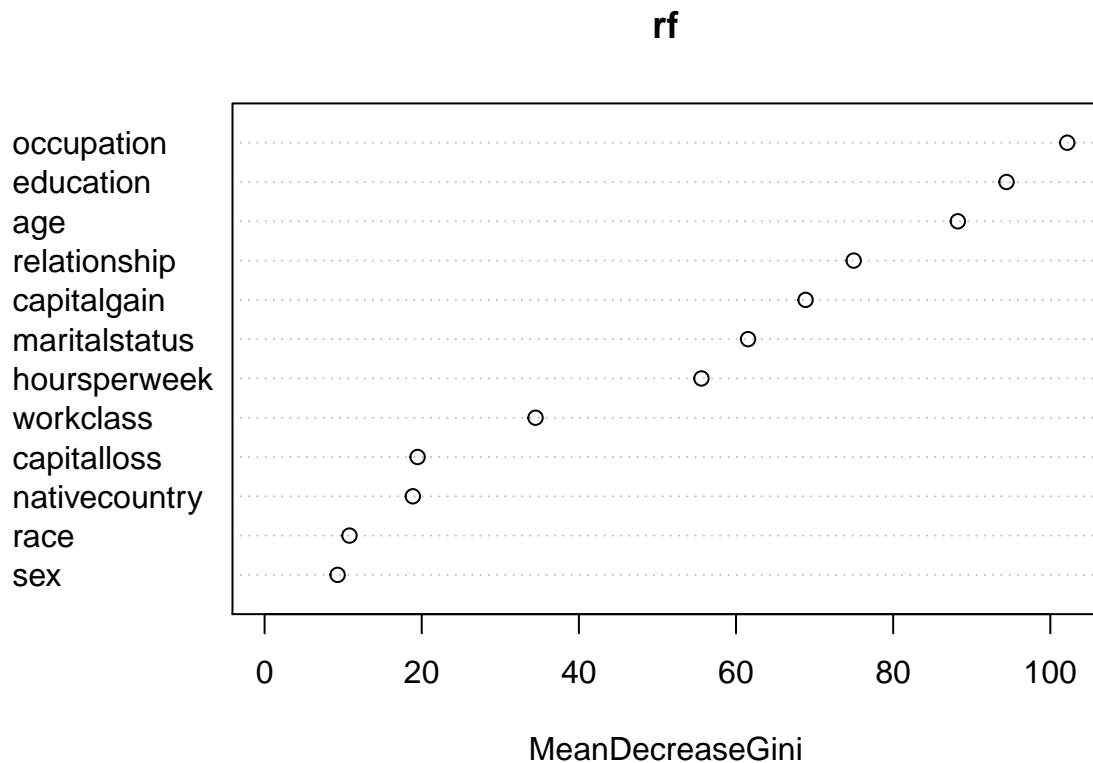
One metric that we can look at is the number of times, aggregated over all of the trees in the random forest model, that a certain variable is selected for a split. To view this metric, run the following lines of R code (replace “MODEL” with the name of your random forest model):

```
vu = varUsed(rf, count=TRUE)
vusorted = sort(vu, decreasing = FALSE, index.return = TRUE)
dotchart(vusorted$x, names(rf$forest$xlevels[vusorted$ix]))
```



A different metric we can look at is related to “impurity”, which measures how homogenous each bucket or leaf of the tree is. In each tree in the forest, whenever we select a variable and perform a split, the impurity is decreased. Therefore, one way to measure the importance of a variable is to average the reduction in impurity, taken over all the times that variable is selected for splitting in all of the trees in the forest. This is evident using:

```
varImpPlot(rf)
```



Finally, we can improve our CART model by tuning its parameters. Let us select the `cp` parameter for our CART model using k-fold cross validation, with $k = 10$ folds. We should test `cp` values from 0.002 to 0.1 in 0.002 increments, by using the following command:

```
#install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.1
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.1
```

```
set.seed(2)
tr.control = trainControl(method="cv", number=10)
cartGrid = expand.grid( .cp = seq(0.002,0.1,0.002))
tr = train(over50k ~., data=train, method="rpart", trControl=tr.control, tuneGrid = cartGrid)
tr
```

```
## CART
```

```
##
```

```
## 19187 samples
```

```
## 12 predictor
```

```
## 2 classes: ' <=50K', ' >50K'
```

```

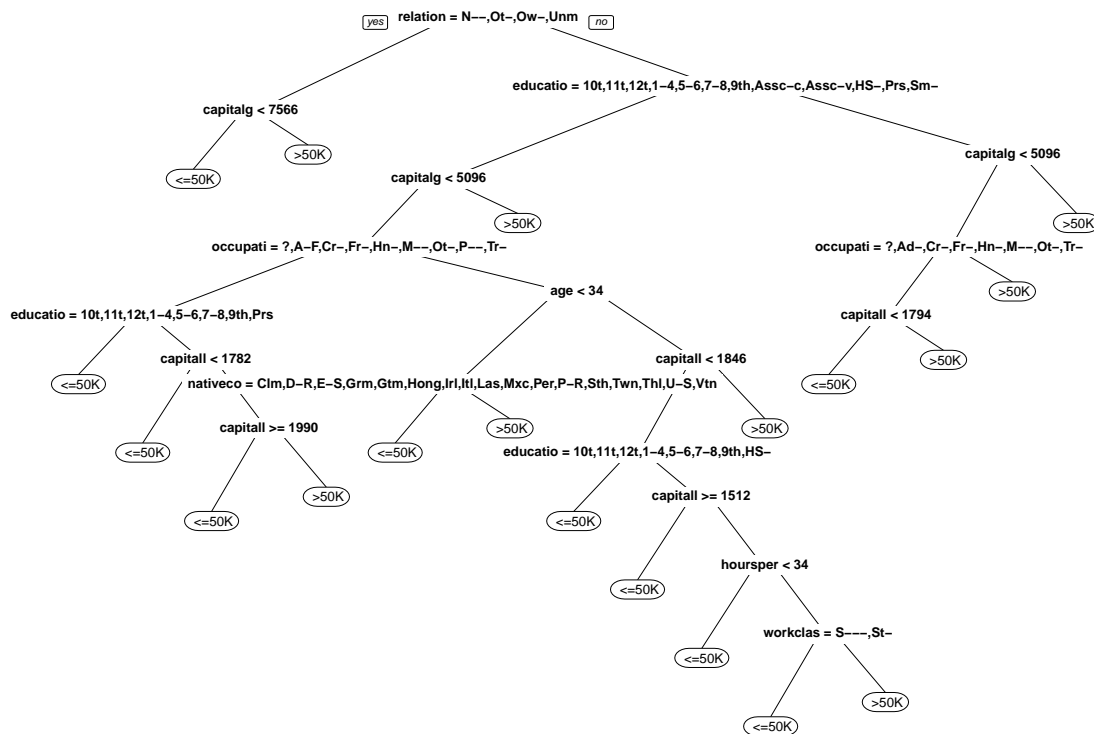
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 17268, 17269, 17268, 17268, 17269, 17269, ...
##
## Resampling results across tuning parameters:
##
##   cp      Accuracy   Kappa      Accuracy SD   Kappa SD
##   0.002  0.8515658  0.55758898  0.0052948552  0.01739909
##   0.004  0.8476566  0.55353089  0.0066966577  0.02355150
##   0.006  0.8444252  0.53388285  0.0065980275  0.02317270
##   0.008  0.8441646  0.53439869  0.0064904574  0.02340138
##   0.010  0.8443210  0.53574983  0.0064991617  0.02413215
##   0.012  0.8443210  0.53574983  0.0064991617  0.02413215
##   0.014  0.8443210  0.53574983  0.0064991617  0.02413215
##   0.016  0.8434351  0.53116984  0.0049698138  0.02002302
##   0.018  0.8405163  0.51397907  0.0063520005  0.03039022
##   0.020  0.8395783  0.50605490  0.0053905706  0.02178219
##   0.022  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.024  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.026  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.028  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.030  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.032  0.8391090  0.50179921  0.0053072840  0.01935313
##   0.034  0.8370243  0.48876941  0.0053526918  0.02398841
##   0.036  0.8321776  0.46362978  0.0062028957  0.02572369
##   0.038  0.8265491  0.43857371  0.0056114951  0.02279967
##   0.040  0.8246203  0.43016472  0.0049425084  0.02374054
##   0.042  0.8246203  0.43016472  0.0049425084  0.02374054
##   0.044  0.8246203  0.43016472  0.0049425084  0.02374054
##   0.046  0.8220148  0.41296653  0.0057721909  0.04769786
##   0.048  0.8220148  0.41296653  0.0057721909  0.04769786
##   0.050  0.8220148  0.41296653  0.0057721909  0.04769786
##   0.052  0.8153954  0.35631451  0.0063204975  0.06119945
##   0.054  0.8128414  0.32562676  0.0062322225  0.05637288
##   0.056  0.8119034  0.30789310  0.0044463297  0.02124615
##   0.058  0.8119034  0.30789310  0.0044463297  0.02124615
##   0.060  0.8119034  0.30789310  0.0044463297  0.02124615
##   0.062  0.8119034  0.30789310  0.0044463297  0.02124615
##   0.064  0.8095585  0.29541138  0.0042248284  0.02501807
##   0.066  0.8095585  0.29541138  0.0042248284  0.02501807
##   0.068  0.8009584  0.24623306  0.0087816673  0.05193321
##   0.070  0.7958509  0.21466597  0.0034057042  0.01841291
##   0.072  0.7958509  0.21466597  0.0034057042  0.01841291
##   0.074  0.7958509  0.21466597  0.0034057042  0.01841291
##   0.076  0.7725557  0.07856024  0.0171457089  0.10171443
##   0.078  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.080  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.082  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.084  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.086  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.088  0.7593684  0.00000000  0.0001912161  0.00000000
##   0.090  0.7593684  0.00000000  0.0001912161  0.00000000

```

```
## 0.092 0.7593684 0.00000000 0.0001912161 0.00000000
## 0.094 0.7593684 0.00000000 0.0001912161 0.00000000
## 0.096 0.7593684 0.00000000 0.0001912161 0.00000000
## 0.098 0.7593684 0.00000000 0.0001912161 0.00000000
## 0.100 0.7593684 0.00000000 0.0001912161 0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002.
```

Obviously, $cp = 0.002$ gives best accuracy. We can now import this parameter into the CART model:

```
CART = rpart(over50k~., data = train, method = "class", cp = 0.002)
prp(CART)
```



```
pred = predict(CART, newdata = test, type="class")
table(test$over50k, pred)
```

```
##      pred
##      <=50K >50K
## <=50K   9178  535
## >50K   1240 1838
```

```
ACU = (9178+1838)/nrow(test)
ACU
```

```
## [1] 0.8612306
```

By tuning `cp`, we improved our accuracy by over 1%, but our tree became significantly more complicated. In some applications, such an improvement in accuracy would be worth the loss in interpretability. In others, we may prefer a less accurate model that is simpler to understand and describe over a more accurate – but more complicated – model.