Income Class Prediction from Census Data

Subsampling and Modeling

In [19]:
```
library(dplyr)
library(ggplot2)
library(lattice)
library(stringr)
library(gridExtra)
library(caret)
library(rpart)
```

The classes in the data set are very imbalanced. The low income class has ~15 times more samples than the high income class. Therefore the low income class will be subsampled. The set of samples with low income will be divided randomly into 15. Each of the subsamples will be combined with the complete high income class data and modeled. Then the 15 models will be ensembled.

First import the complete data after imputation.

In [2]:
```
y <- read.csv('C:/Datasets/censusincomedfFull.csv')$income
X <- read.csv('C:/Datasets/censusincomeXFull.csv')
X <- as.matrix(X[,-1])
```

All samples are randomly assigned a number between 1 and 15, using the following vector.

In [3]:
```
subsetnumber <- floor(runif(length(y),1,16))
```

Model Tuning

Different models will be explored with the first subset.

In [4]:
```
y1 <- y[(y==' 50000+.')|(subsetnumber==1)]
X1 <- X[(y==' 50000+.')|(subsetnumber==1),]
```

The nature of the data is such that the relevance of many features is dependent on the levels of other features. Some levels of some features will always be associated with 'Not in universe' for other features. In other words, not all combinations of levels of features is possible. Therefore tree models are more appropriate for the data than models based on n-dimensional spaces (kNN, SVM, LDA etc.). Tree models do not require the scaling of variables.

Due to the large size of the data sets, a simple decision tree model will be used instead of random forest or boosted trees. Cross validation will be used to optimize the complexity parameter to avoid overfitting. Variance will be further reduced when all 15 models are ensembled at the end. Model tuning will be performed on the first subset of the data, and the resulting complexity parameter will be applied to the other subsets.

```
In [33]: Tree1 <- train(X1,y1,method ='rpart',trControl=trainControl(method='repeatedcv',numb
         Tree1
```

```
CART 

24680 samples
  470 predictor
    2 classes: ' - 50000.', ' 50000+.' 

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times) 
Summary of sample sizes: 19745, 19743, 19744, 19745, 19743, 19744, ... 
Resampling results across tuning parameters:

  cp          Accuracy   Kappa    
  0.01784843  0.8010539  0.6019296
  0.05773296  0.7821583  0.5639969
  0.52764677  0.6587318  0.3158086

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01784843.
```

```
In [6]: Grid <- expand.grid(cp=c(0.01,0.012,0.014,0.016,0.018,0.02))
        Tree1 <- train(X1,y1,method ='rpart',tuneGrid=Grid,trControl=trainControl(method='re
        Tree1
```

```
CART 

24556 samples
  470 predictor
    2 classes: ' - 50000.', ' 50000+.' 

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times) 
Summary of sample sizes: 19645, 19644, 19645, 19645, 19645, 19645, ... 
Resampling results across tuning parameters:

  cp     Accuracy   Kappa    
  0.010  0.8177903  0.6351486
  0.012  0.8150483  0.6297254
  0.014  0.8115461  0.6228002
  0.016  0.8085190  0.6167348
  0.018  0.8063065  0.6123177
  0.020  0.7931662  0.5858328

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01.
```

In [7]:
```
Grid <- expand.grid(cp=c(0.002,0.004,0.006,0.008, 0.01))
Tree1 <- train(X1,y1,method ='rpart',tuneGrid=Grid,trControl=trainControl(method='re
Tree1
```

```
CART

24556 samples
  470 predictor
    2 classes: ' - 50000.', ' 50000+.'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 19645, 19646, 19644, 19644, 19645, 19645, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  0.002  0.8397404  0.6791669
  0.004  0.8339711  0.6676271
  0.006  0.8268989  0.6534380
  0.008  0.8198271  0.6392343
  0.010  0.8165964  0.6327686

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.002.
```

In [8]:
```
Grid <- expand.grid(cp=c(0.0002,0.0004,0.0006,0.0008, 0.001))
Tree1 <- train(X1,y1,method ='rpart',tuneGrid=Grid,trControl=trainControl(method='re
Tree1
```

```
CART

24556 samples
  470 predictor
    2 classes: ' - 50000.', ' 50000+.'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 19646, 19645, 19644, 19645, 19644, 19646, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  2e-04  0.8533558  0.7065983
  4e-04  0.8541161  0.7080684
  6e-04  0.8528537  0.7055255
  8e-04  0.8503967  0.7006144
  1e-03  0.8485369  0.6968867

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 4e-04.
```

In [9]:
```
Grid <- expand.grid(cp=c(0.0002,0.0003,0.0004,0.0005))
Tree1 <- train(X1,y1,method ='rpart',tuneGrid=Grid,trControl=trainControl(method='re
Tree1
```

```
CART

24556 samples
  470 predictor
    2 classes: ' - 50000.', ' 50000+.'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 3 times)
Summary of sample sizes: 19645, 19646, 19645, 19644, 19644, 19644, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  2e-04  0.8509259  0.7017316
  3e-04  0.8541297  0.7081031
  4e-04  0.8546048  0.7090343
  5e-04  0.8538581  0.7075450

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 4e-04.
```

The complexity parameter value of 0.0004 will be used to build the models.

Model Fitting

The remaining 14 models are fit below.

In [28]:
```
for(i in 2:15){
    Xtemp <- as.data.frame(X[(y==' 50000+.')|(subsetnumber==i),])
    Xtemp$y <-y[(y==' 50000+.')|(subsetnumber==i)]
    assign(paste('Tree',i,sep=''),rpart(y~.,Xtemp,cp=0.0004))
}
```

Prediction and Ensembling

Below is a function that will generate the ensembled prediction.

In [46]:
```r
ensemblepredict <- function(predictors){
    for(i in 1:15){
        if(i==1){
            ypred <- as.data.frame(predict(get(paste('Tree',i,sep='')),predictors))
            names(ypred)=c('y1')
        }
        else{
            ypred[,paste('y',i,sep='')] <- predict(get(paste('Tree',i,sep='')),predi
        }
    }
    return(ypred)
}
```

Predicting entire data set.

In [47]:
```r
yfullpred <- ensemblepredict(as.data.frame(X))
head(yfullpred,5)
```

y1	y2	y3	y4	y5	y6	y7	y8	y9	y10	y11	y12	
-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	
50000+.	50000+.	-50000.	-50000.	50000+.	-50000.	50000+.	50000+.	-50000.	50000+.	-50000.	50000+.	5
-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	
-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	
-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	-50000.	

The majority class from the 15 predictions is taken as the final prediction. The results are summarized below.

```
In [58]: ypred <- factor(apply(yfullpred,1,function(x){x[which.max(table(x))]}))
         confusionMatrix(ypred, y, positive =' 50000+.')
```

```
Confusion Matrix and Statistics

           Reference
Prediction   - 50000.  50000+.
   - 50000.    152152     1295
   50000+.      31760    11087

               Accuracy : 0.8316
                 95% CI : (0.8299, 0.8333)
    No Information Rate : 0.9369
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3366
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.89541
            Specificity : 0.82731
         Pos Pred Value : 0.25876
         Neg Pred Value : 0.99156
             Prevalence : 0.06308
         Detection Rate : 0.05648
   Detection Prevalence : 0.21828
      Balanced Accuracy : 0.86136

       'Positive' Class :  50000+.
```

The final model is able to predict most of each class correctly, however it over predicts the positive class by a factor of 3, which results in very low precision. This is mainly due to the imbalance in the data. Since there are so many more instances of the negative class, even a small percentage of them that are misclassified is a large in comparison to the number of instances of the positive class. This likely would not have been as extreme if the data were not subsampled to balance the classes during model fitting. However, leaving the classes unbalanced would have resulted in under prediction of the positive class. We see that the balanced accuracy is higher than the accuracy value, indicating that the model would perform better on a balanced data set.

Aside from the impact of imbalance, the high proportion of false positives could also be due to the fact that there may be other features not in the data set that contribute to a person's income. Success may depend highly on chance and personality in addition to the background/demographics of a person. Therefore only a minority of people with a certain background that is most associated with success may actually be able to acheive the high income.