AMRC: Exercise 10 - 11912007

Analysis

In this exercise, we aimed to predict caravan insurance policy purchases using classification trees and random forests. We paid special attention to the challenge of imbalanced classes, where the “Yes” purchasers are in the minority.

Note: Despite our best efforts to ensure reproducibility by setting a seed (42), the results may vary slightly each time the analysis is run. We do not know why this is the case.

For the classification trees analysis, an initial tree T0 was created using the `rpart` function with balanced prior probabilities (0.5, 0.5) to avoid bias towards the majority class. The initial model achieved a balanced accuracy of 0.67, with the confusion matrix showing 69 correct positive predictions and 1353 correct negative predictions. After cross-validation analysis, the optimal complexity parameter was determined to be 0.021, which was used to prune the initial tree. The pruned tree showed improved performance with a balanced accuracy of 0.718, demonstrating that pruning helped reduce overfitting.

An attempt to improve performance using weighted trees was made by assigning weights inversely proportional to class frequencies. However, this approach yielded the same balanced accuracy of 0.67 as the initial tree, suggesting that the weighting strategy did not provide additional benefits in this case.

The random forest analysis began with a basic model using 100 trees to prevent memory issues. The initial random forest model showed poor performance with a balanced accuracy of 0.536. Three strategies were then tested to improve performance: undersampling the majority class using `sampsize`, assigning higher weights to the minority class using `classwt`, and adjusting the prediction threshold using `cutoff`.

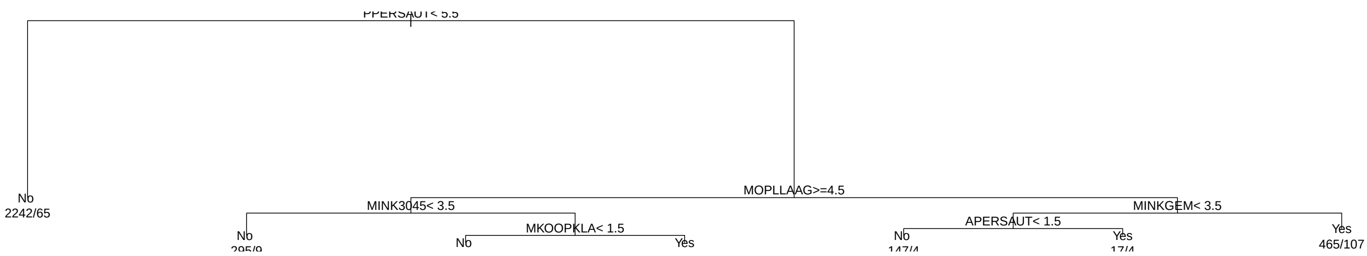
Among these strategies, the weighted random forest approach (classwt) proved most effective, achieving the highest balanced accuracy of 0.607. This model assigned five times more weight to the minority class (`classwt=c(1,5)`), helping to counteract the class imbalance. The cutoff adjustment strategy was the second-best performer with a balanced accuracy of 0.593.

The variable importance plots from the best model (weighted random forest) revealed the most influential predictors for classification.

Code

Decision Trees

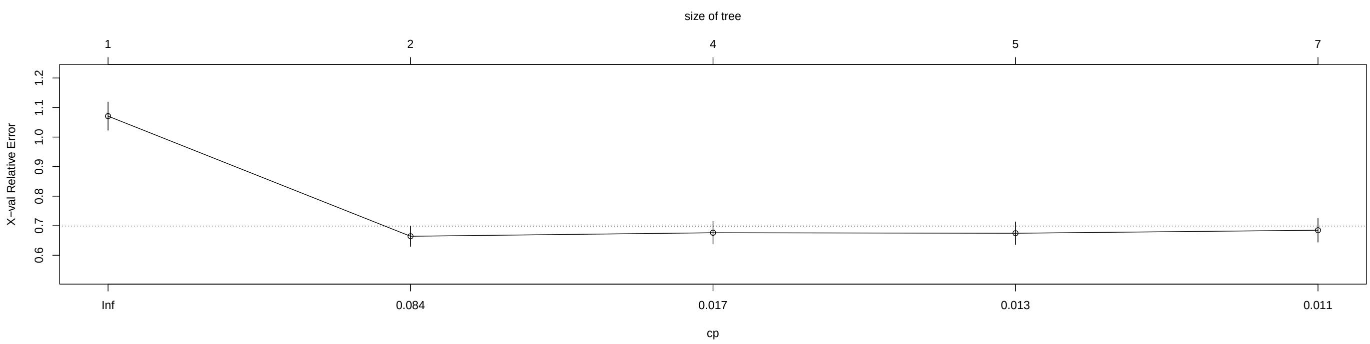
```r
train_idx <- sample(1:nrow(Caravan), size = round(nrow(Caravan) * 2/3))
train_data <- Caravan[train_idx, ]
test_data <- Caravan[-train_idx, ]

#
# initial tree
#

# (a) compute initial tree T0
# 50/50 prior avoids bias towards majority class
tree_0 <- rpart(Purchase ~ ., data = train_data, method = "class", parms = list(prior = c(0.5, 0.5)))

# (b) visualize T0
plot(tree_0)
text(tree_0, use.n = TRUE)
```




```r
# (c) eval, confusion table, balanced accuracy
eval_tree <- function(model, test_data, model_name = "model") {
    pred_test <- predict(model, test_data, type = "class")
    conf_matrix <- table(Actual = test_data$Purchase, Predicted = pred_test)

    sensitivity <- conf_matrix[2,2] / sum(conf_matrix[2,])
    specificity <- conf_matrix[1,1] / sum(conf_matrix[1,])
    bal_acc <- (sensitivity + specificity) / 2

    cat("confusion matrix (", model_name, "):\n", sep="")
    print(conf_matrix)
    cat("balanced accuracy (", model_name, "): ", bal_acc, "\n", sep="")
}
eval_tree(tree_0, test_data, "T0")
```

```
## confusion matrix (T0):
##       Predicted
## Actual   No  Yes
##    No  1353  473
##    Yes   46   69
## balanced accuracy (T0): 0.6704819
```

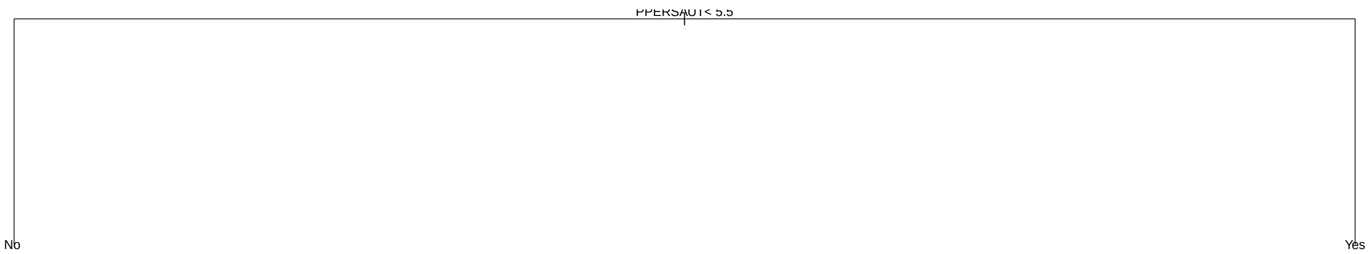
```r
# (d) cross-validation, optimal complexity
plotcp(tree_0)
```



```r
opt_cp <- tree_0$cptable[which.min(tree_0$cptable[,"xerror"]), "CP"]
cat("optimal complexity:", opt_cp, "\n")
```

```
## optimal complexity: 0.02111981
```

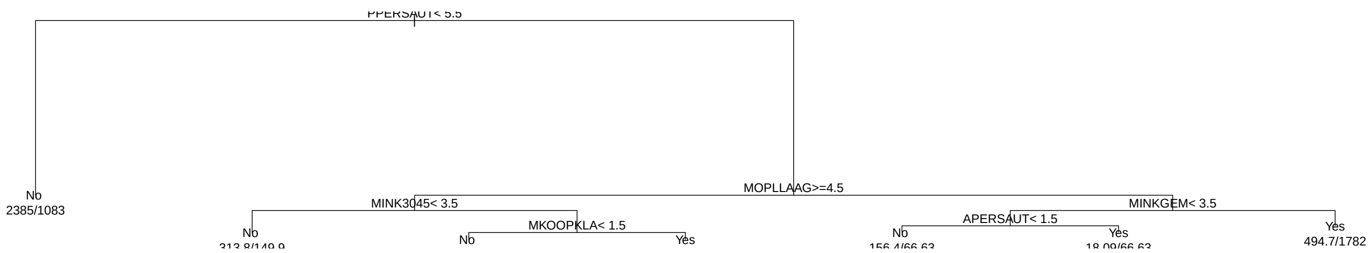
```r
#
# pruned tree
#

# (e) prune tree
tree_pruned <- prune(tree_0, cp = opt_cp)
plot(tree_pruned)
text(tree_pruned, use.n = TRUE)
```




```r
# (f) eval pruned tree
eval_tree(tree_pruned, test_data, "pruned tree")
```

```
## confusion matrix (pruned tree):
##       Predicted
## Actual   No  Yes
##    No  1131  695
##    Yes   21   94
## balanced accuracy (pruned tree): 0.718389
```

```r
#
# weighted tree
#

# (g) weighted tree for imbalanced classes
# weights inversely proportional to class frequencies
weights <- ifelse(train_data$Purchase == "Yes", 1/mean(train_data$Purchase == "Yes"), 1/mean(train_data$Purchase == "No"))
weighted_tree <- rpart(Purchase ~ ., data = train_data, method = "class", weights = weights)

plot(weighted_tree)
text(weighted_tree, use.n = TRUE)
```

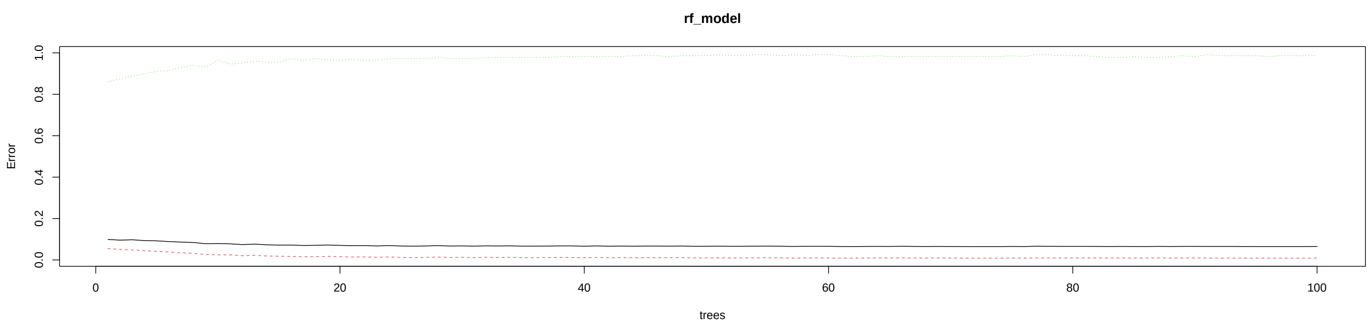



```r
eval_tree(weighted_tree, test_data, "weighted tree")
```

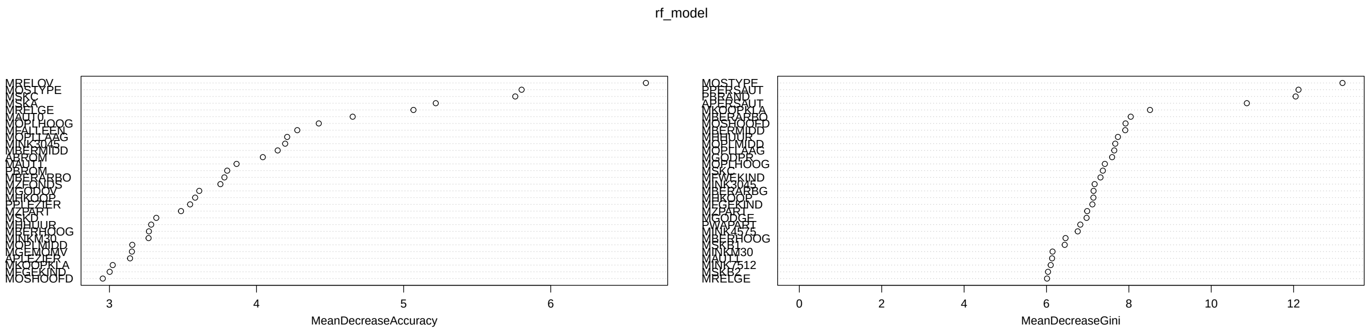
```
## confusion matrix (weighted tree):
##       Predicted
## Actual   No  Yes
##    No  1353  473
##    Yes   46   69
## balanced accuracy (weighted tree): 0.6704819
```

Random Forest

```r
train_idx <- sample(1:nrow(Caravan), size = round(nrow(Caravan) * 2/3))
train_data <- Caravan[train_idx, ]
test_data <- Caravan[-train_idx, ]

ntrees = 100 # avoid out of memory errors

# (a) random forest
rf_model <- randomForest(Purchase ~ ., data = train_data, importance = TRUE, ntree = ntrees)

# (b) plot
plot(rf_model)
```



```r
varImpPlot(rf_model)
```



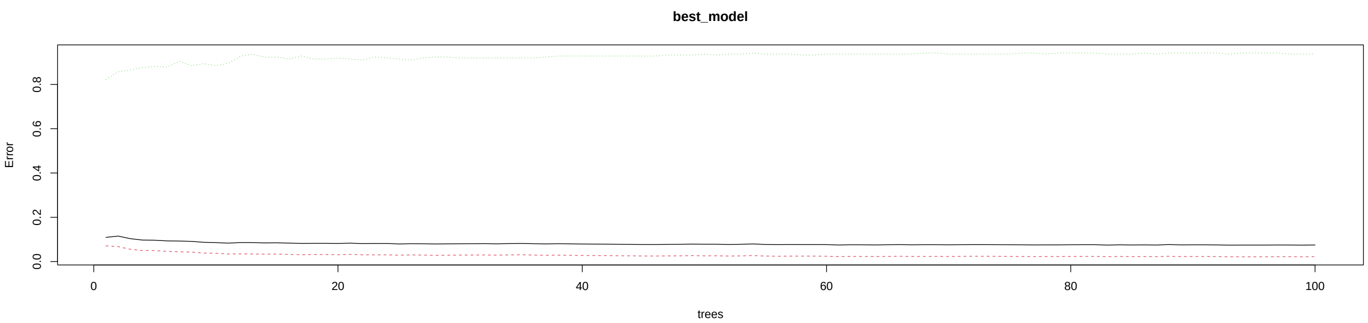

```r
# (c1) sampsize: undersample majority class
majority_size <- sum(train_data$Purchase == "No")
minority_size <- sum(train_data$Purchase == "Yes")
sample_size <- c(majority_size, minority_size)
rf_balanced_samp <- randomForest(Purchase ~ ., data = train_data, importance = TRUE, ntree = ntrees, sampsize = sample_size)

# (c2) classwt: assign higher weight to minority class
rf_weighted <- randomForest(Purchase ~ ., data = train_data, importance = TRUE, ntree = ntrees, classwt = c(1, 5))

# (c3) cutoff: lower threshold for predicting minority class
rf_cutoff <- randomForest(Purchase ~ ., data = train_data, importance = TRUE, ntree = ntrees, cutoff = c(0.7, 0.3))

# (d) best strategy
eval_ensemble <- function(model, name) {
    preds <- predict(model, newdata = test_data)
    conf_matrix <- table(test_data$Purchase, preds)
    cat("confusion matrix (", name, "):\n", sep="")
    print(conf_matrix)
    bal_acc <- confusionMatrix(conf_matrix)$byClass["Balanced Accuracy"]
    cat("balanced accuracy (", name, "):", bal_acc, "\n", sep="")
    return(bal_acc)
}
experiments <- list(rf_model, rf_balanced_samp, rf_weighted, rf_cutoff)
names <- c("rf_model", "rf_balanced_samp", "rf_weighted", "rf_cutoff")
results <- sapply(1:length(experiments), function(i) eval_ensemble(experiments[[i]], names[i]))
```

```
## confusion matrix (rf_model):
##      preds
##         No  Yes
##   No  1797   19
##   Yes  122    3
## balanced accuracy (rf_model):0.5363944
## confusion matrix (rf_balanced_samp):
##      preds
##         No  Yes
##   No  1795   21
##   Yes  122    3
## balanced accuracy (rf_balanced_samp):0.5306794
## confusion matrix (rf_weighted):
##      preds
##         No  Yes
##   No  1784   32
##   Yes  113   12
## balanced accuracy (rf_weighted):0.6065798
## confusion matrix (rf_cutoff):
##      preds
##         No  Yes
##   No  1760   56
##   Yes  107   18
## balanced accuracy (rf_cutoff):0.592966
```

```r
best_model <- experiments[[which.max(results)]]
best_name <- names[which.max(results)]

cat("best model:", best_name, "\n")
```

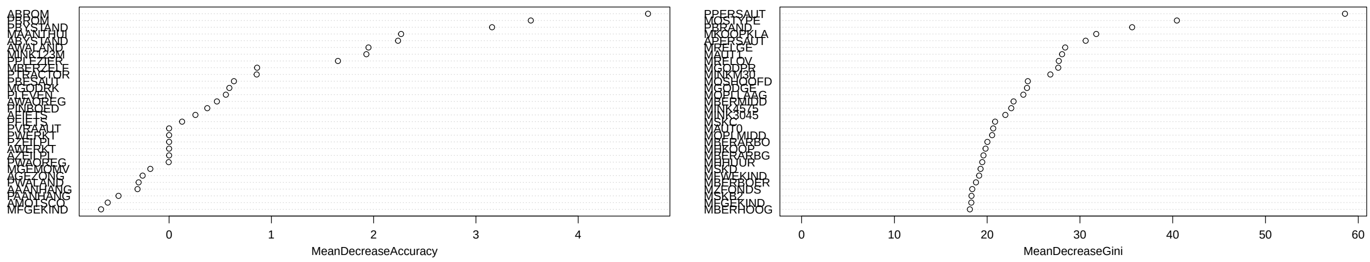
```
## best model: rf_weighted
```

```r
plot(best_model)
```




```r
varImpPlot(best_model)
```

best_model



ABROM
PBROM
PBYSTAND
MAANTHUI
ABYSTAND
AWALAND
MINK123M
PPLEZIER
MBERZELF
PTRACTOR
PBESAUT
MGODRK
PLEVEN
AWAOREG
PINBOED
AFIETS
PFIETS
PVRAAUT
PWERKT
PZEILPL
AWERKT
AZEILPL
PWAOREG
MGEMOMV
AGEZONG
PWALAND
AAANHANG
PAANHANG
AMOTSCO
MFGEKIND
0
1
2
3
4
MeanDecreaseAccuracy
PPERSAUT
MOSTYPE
PBRAND
MKOOPKLA
APERSAUT
MRELGE
MAUT1
MRELOV
MGODPR
MINKM30
MOSHOOFD
MGODGE
MOPLLAAG
MBERMIDD
MINK4575
MINK3045
MSKC
MAUT0
MOPLMIDD
MBERARBO
MHKOOP
MBERARBG
MHHUUR
MSKD
MFWEKIND
MBERBOER
MZFONDS
MSKB2
MFGEKIND
MBERHOOG
0
10
20
30
40
50
60
MeanDecreaseGini