Decision_Tree

```
## Loading required package: lattice
## Loading required package: ggplot2
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

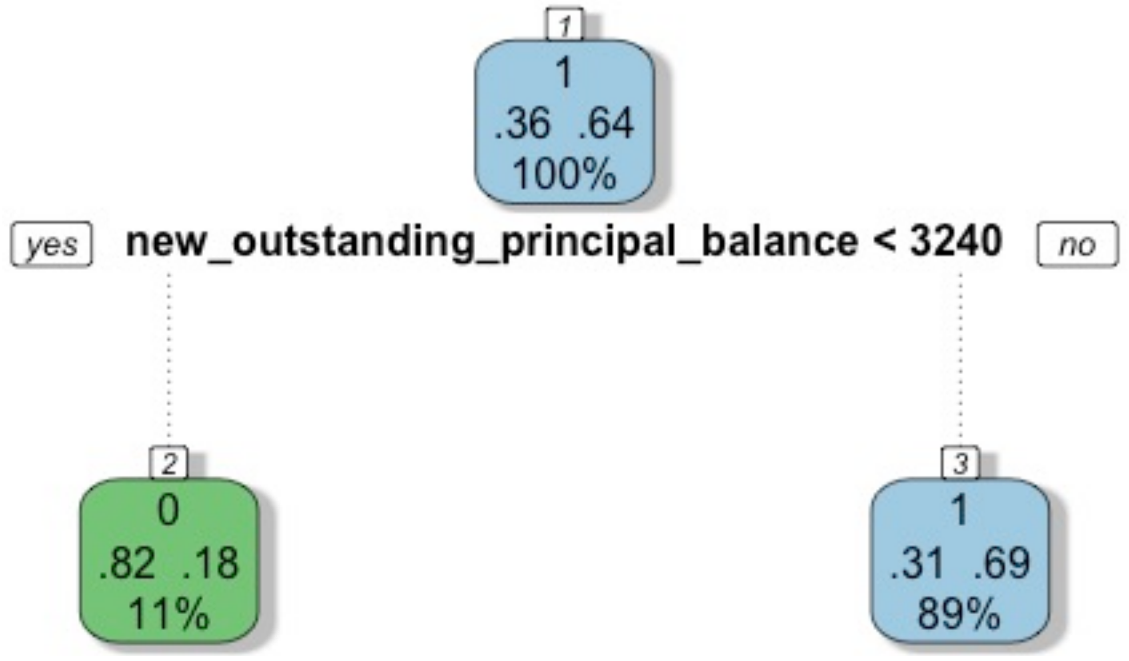
Stratified sampling

```
set.seed(123456)
inTrain = createDataPartition(y=data$target, p=0.75, list=FALSE)
training = data[inTrain,]
testing = data[-inTrain,]
dim(training);dim(testing)

## [1] 358  19

## [1] 119  19
```

Decision Tree

```
formular1 = as.factor(target)~new_outstanding_principal_balance+initial_loan_amount
+fico+sales_channel__c+type+current_collection_method+term+average_bank_balance__c+
lender1


my_tree <- train(formular1,data=training,method = "rpart",cp=0.06)
```

```r
fancyRpartPlot(my_tree$finalModel)
```



```r
print(my_tree)
```

```
## CART
##
## 358 samples
##  18 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 358, 358, 358, 358, 358, 358, ...
```

```
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.02325581  0.6381900  0.1810894  0.05542267   0.1065945
##   0.02583979  0.6432728  0.1899243  0.05035546   0.0916527
##   0.18604651  0.6608524  0.1198283  0.04275220   0.1011020
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.1860465.
```

Apply decision tree to the testing data

```
tree_pred <- predict(my_tree, testing)
tree_pred_df <- data.frame(tree_pred, target=testing$target)

confusionTree=table(tree_pred_df$tree_pred, tree_pred_df$target)
print(confusionTree)
```

```
##
##      0  1
##   0  0  0
##   1 34 85
```

```
Observed_Accuracy = (confusionTree[1,1]+confusionTree[2,2])/sum(confusionTree)
print(Observed_Accuracy)
```

```
## [1] 0.7142857
```

```
Expected_Accuracy = (sum(confusionTree[,1])*sum(confusionTree[1,])/sum(confusionTre
e)+sum(confusionTree[,2])*sum(confusionTree[2,])/sum(confusionTree))/sum(confusionT
ree)

kappa = (Observed_Accuracy-Expected_Accuracy)/(1-Expected_Accuracy)
print(kappa)
```

```
## [1] 0
```