Tree Model

Author: Rajat Jain

Last Updated: 2018-04-24

Contents

Training & Test Data

We have split available usage data into training data (75% - 1987 records) and test data(25% - 663 records).

Summary of Training data

```
##           class       lr_cc_usage       lr_cl_usage       lr_mo_usage
##  OTHER       :1106   Min.   : 0.0000   Min.   :  0.000   Min.   : 0.0000
##  PHOTOGRAPHER: 881   1st Qu.: 0.0000   1st Qu.:  0.000   1st Qu.: 0.0000
##                      Median : 0.0000   Median :  2.000   Median : 0.0000
##                      Mean   : 0.3563   Mean   :  4.265   Mean   : 0.8938
##                      3rd Qu.: 0.0000   3rd Qu.:  6.000   3rd Qu.: 0.0000
##                      Max.   :20.0000   Max.   :185.000   Max.   :24.0000
##  storage_usage         ps_usage        stock_usage
##  Min.   :     0.0   Min.   :  0.000   Min.   :  0.000
##  1st Qu.:     0.0   1st Qu.:  0.000   1st Qu.:  0.000
##  Median :     0.0   Median :  3.000   Median :  0.000
##  Mean   :   255.3   Mean   :  4.703   Mean   :  1.099
##  3rd Qu.:     1.0   3rd Qu.:  6.000   3rd Qu.:  0.000
##  Max.   :107556.0   Max.   :182.000   Max.   :246.000
```

Summary of Test data

```
##           class      lr_cc_usage       lr_cl_usage      lr_mo_usage
##  OTHER       :394   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  PHOTOGRAPHER:269   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##                     Median : 0.0000   Median : 2.000   Median : 0.0000
##                     Mean   : 0.3213   Mean   : 4.072   Mean   : 0.7587
##                     3rd Qu.: 0.0000   3rd Qu.: 6.000   3rd Qu.: 0.0000
##                     Max.   :22.0000   Max.   :81.000   Max.   :21.0000
##  storage_usage      ps_usage       stock_usage
##  Min.   :    0   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.:    0   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median :    0   Median : 2.000   Median : 0.0000
##  Mean   :  436   Mean   : 4.487   Mean   : 0.7104
##  3rd Qu.:    1   3rd Qu.: 6.000   3rd Qu.: 0.0000
##  Max.   :96273   Max.   :92.000   Max.   :48.0000
```

Training - Tree Model

As the next step, we build a simple decision tree classifier model.

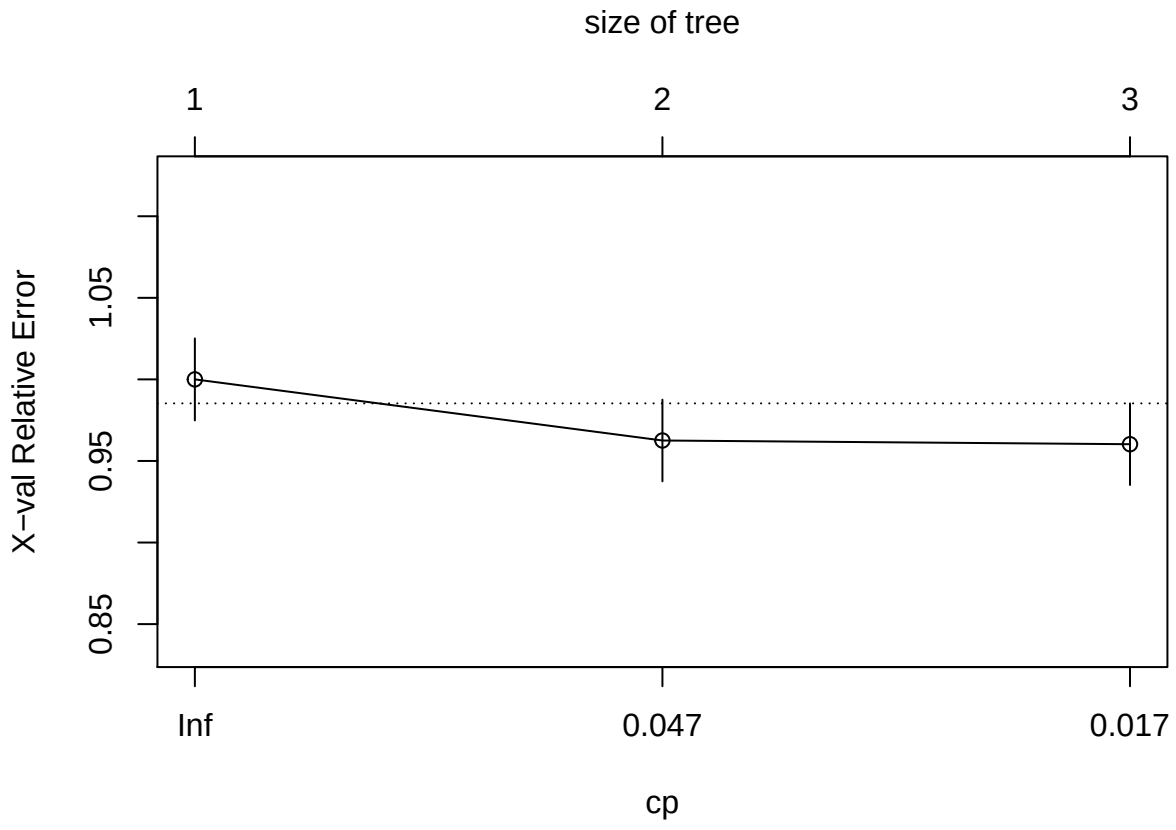
```
# grow tree
model <- rpart(class ~ lr_cc_usage + lr_cl_usage + storage_usage + ps_usage + stock_usage,
               method="class", data=train)
```

Summarize trained model.

```
printcp(model) # display the results
```

```
##
## Classification tree:
## rpart(formula = class ~ lr_cc_usage + lr_cl_usage + storage_usage +
##     ps_usage + stock_usage, data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] lr_cl_usage ps_usage
##
## Root node error: 881/1987 = 0.44338
##
## n= 1987
##
##         CP nsplit rel error  xerror     xstd
## 1 0.074915      0   1.00000 1.00000 0.025136
## 2 0.029512      1   0.92509 0.96254 0.025026
## 3 0.010000      2   0.89557 0.96027 0.025018
```

```
plotcp(model) # visualize cross-validation results
```



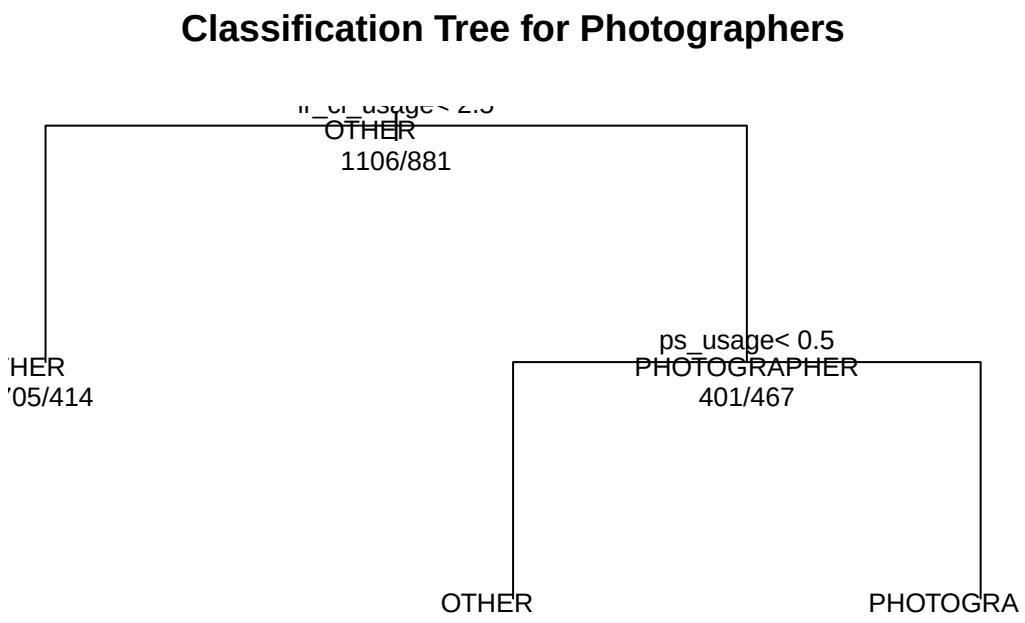
```r
summary(model) # detailed summary of splits
```

```
## Call:
## rpart(formula = class ~ lr_cc_usage + lr_cl_usage + storage_usage +
##     ps_usage + stock_usage, data = train, method = "class")
##   n= 1987
##
##           CP nsplit rel error    xerror       xstd
## 1 0.07491487      0 1.0000000 1.0000000 0.02513568
## 2 0.02951192      1 0.9250851 0.9625426 0.02502563
## 3 0.01000000      2 0.8955732 0.9602724 0.02501803
##
## Variable importance
##   lr_cl_usage      ps_usage storage_usage
##            77            20             3
##
## Node number 1: 1987 observations,    complexity param=0.07491487
##   predicted class=OTHER         expected loss=0.443382  P(node) =1
##     class counts:  1106   881
##    probabilities: 0.557 0.443
##   left son=2 (1119 obs) right son=3 (868 obs)
##   Primary splits:
##       lr_cl_usage   < 2.5 to the left,  improve=27.6079600, (0 missing)
##       lr_cc_usage   < 0.5 to the right, improve= 6.9303420, (0 missing)
##       ps_usage      < 8.5 to the left,  improve= 2.9357070, (0 missing)
##       storage_usage < 0.5 to the right, improve= 2.1032180, (0 missing)
##       stock_usage   < 31  to the left,  improve= 0.4729293, (0 missing)
##   Surrogate splits:
##       storage_usage < 71  to the left,  agree=0.583, adj=0.045, (0 split)
##       ps_usage      < 0.5 to the right, agree=0.581, adj=0.041, (0 split)
##
## Node number 2: 1119 observations
##   predicted class=OTHER         expected loss=0.3699732  P(node) =0.5631605
##     class counts:   705   414
##    probabilities: 0.630 0.370
##
## Node number 3: 868 observations,    complexity param=0.02951192
##   predicted class=PHOTOGRAPHER  expected loss=0.4619816  P(node) =0.4368395
##     class counts:   401   467
##    probabilities: 0.462 0.538
##   left son=6 (274 obs) right son=7 (594 obs)
##   Primary splits:
##       ps_usage      < 0.5 to the left,  improve=5.8489390, (0 missing)
##       lr_cc_usage   < 0.5 to the right, improve=5.1682220, (0 missing)
##       lr_cl_usage   < 9.5 to the left,  improve=4.2876010, (0 missing)
##       storage_usage < 0.5 to the right, improve=3.6656130, (0 missing)
##       stock_usage   < 3.5 to the left,  improve=0.6957295, (0 missing)
##   Surrogate splits:
##       lr_cc_usage < 10  to the right, agree=0.685, adj=0.004, (0 split)
##
## Node number 6: 274 observations
##   predicted class=OTHER         expected loss=0.4525547  P(node) =0.1378963
##     class counts:   150   124
##    probabilities: 0.547 0.453
```

```
##
## Node number 7: 594 observations
##   predicted class=PHOTOGRAPHER  expected loss=0.4225589  P(node) =0.2989431
##     class counts:   251   343
##    probabilities: 0.423 0.577
```

Plot tree.

```
plot(model, uniform=TRUE,
     main="Classification Tree for Photographers")
text(model, use.n=TRUE, all=TRUE, cex=.8)
```



Prediction (Testing)

Once we have the model built on the training data, let's test in by predicting the output class on the test data.

```
pred <- predict(model, newdata=test, type="class")
```

Performance

Based on the measure defined in the FPS, we will use classification accuracy as our performance measure.

Confusion Matrix

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction     OTHER PHOTOGRAPHER
##   OTHER          298          165
##   PHOTOGRAPHER    96          104
##
##                Accuracy : 0.6063
```

```
##                  95% CI : (0.568, 0.6437)
##     No Information Rate : 0.5943
##     P-Value [Acc > NIR] : 0.2771
##
##                   Kappa : 0.149
##  Mcnemar's Test P-Value : 2.564e-05
##
##             Sensitivity : 0.3866
##             Specificity : 0.7563
##          Pos Pred Value : 0.5200
##          Neg Pred Value : 0.6436
##              Prevalence : 0.4057
##          Detection Rate : 0.1569
##    Detection Prevalence : 0.3017
##       Balanced Accuracy : 0.5715
##
##        'Positive' Class : PHOTOGRAPHER
##
```

Accuracy

- Observed Accuracy : 60.63%
- Desired accuracy : 70%
- Performance is Not Satisfactory.