German Bank Credit

Decision tree and Random Forest Algorithms

Sean O’Malley

Objective : Minimize loss from the bank’s perspective.

Methodology 1 : Decision Tree

Methodology 2 : Random Forest

Ingest and Explore Data

```
bank <- read.delim("https://onlinecourses.science.psu.edu/stat857/sites/onlinecourses.science.psu.
edu.stat857/files/german_credit.csv", sep = ",", header = T)

# Data Type manipulation

bank <- as.data.frame(mapply(as.factor, bank))

bank$Duration.in.Current.address <- as.integer(bank$Duration.in.Current.address)
bank$Credit.Amount <- as.numeric(bank$Credit.Amount)
bank$Age..years. <- as.integer(bank$Age..years.)
bank$Duration.of.Credit..month. <- as.integer(bank$Duration.of.Credit..month.)
bank$No.of.Credits.at.this.Bank <- as.integer(bank$No.of.Credits.at.this.Bank)
bank$Instalment.per.cent <- as.numeric(bank$Instalment.per.cent)


glimpse(bank)
```

```
## Observations: 1,000
## Variables: 21
## $ Creditability                      <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Account.Balance                    <fctr> 1, 1, 2, 1, 1, 1, 1, 1, 4, ...
## $ Duration.of.Credit..month.         <int> 8, 33, 3, 3, 3, 1, 32, 28, 8...
## $ Payment.Status.of.Previous.Credit <fctr> 4, 4, 2, 4, 4, 4, 4, 4, 4, ...
## $ Purpose                            <fctr> 2, 0, 9, 0, 0, 0, 0, 0, 3, ...
## $ Credit.Amount                      <dbl> 15, 463, 867, 331, 344, 356,...
## $ Value.Savings.Stocks               <fctr> 1, 1, 2, 1, 1, 1, 1, 1, 1, ...
## $ Length.of.current.employment       <fctr> 2, 3, 4, 3, 3, 2, 4, 2, 1, ...
## $ Instalment.per.cent                <dbl> 4, 2, 2, 3, 4, 1, 1, 2, 4, 1...
## $ Sex...Marital.Status               <fctr> 2, 3, 2, 3, 3, 3, 3, 3, 2, ...
## $ Guarantors                         <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Duration.in.Current.address        <int> 4, 2, 4, 2, 4, 3, 4, 4, 4, 4...
## $ Most.valuable.available.asset      <fctr> 2, 1, 1, 1, 2, 1, 1, 1, 3, ...
## $ Age..years.                        <int> 3, 18, 5, 21, 20, 30, 21, 22...
## $ Concurrent.Credits                 <fctr> 3, 3, 3, 3, 1, 3, 3, 3, 3, ...
## $ Type.of.apartment                  <fctr> 1, 1, 1, 1, 2, 1, 2, 2, 2, ...
## $ No.of.Credits.at.this.Bank         <int> 1, 2, 1, 2, 2, 2, 2, 1, 2, 1...
## $ Occupation                         <fctr> 3, 3, 2, 2, 2, 2, 2, 2, 1, ...
## $ No.of.dependents                   <fctr> 1, 2, 1, 2, 1, 2, 1, 2, 1, ...
## $ Telephone                          <fctr> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Foreign.Worker                     <fctr> 1, 1, 1, 2, 2, 2, 2, 2, 1, ...
```

```
summary(bank)
```

```
##  Creditability Account.Balance Duration.of.Credit..month.
##  0:300         1:274           Min.   : 1.00
##  1:700         2:269           1st Qu.: 6.00
##                3: 63           Median :12.00
##                4:394           Mean   :13.34
##                                3rd Qu.:18.00
##                                Max.   :33.00
##
##  Payment.Status.of.Previous.Credit    Purpose    Credit.Amount
##  0: 40                              3      :280   Min.   :  1.0
##  1: 49                              0      :234   1st Qu.:211.8
##  2:530                              2      :181   Median :444.5
##  3: 88                              1      :103   Mean   :450.9
##  4:293                              9      : 97   3rd Qu.:683.2
##                                     6      : 50   Max.   :923.0
##                                     (Other): 55
##  Value.Savings.Stocks Length.of.current.employment Instalment.per.cent
##  1:603                1: 62                        Min.   :1.000
##  2:103                2:172                        1st Qu.:2.000
##  3: 63                3:339                        Median :3.000
##  4: 48                4:174                        Mean   :2.973
##  5:183                5:253                        3rd Qu.:4.000
##                                                    Max.   :4.000
##
##  Sex...Marital.Status Guarantors Duration.in.Current.address
##  1: 50                1:907      Min.   :1.000
##  2:310                2: 41      1st Qu.:2.000
##  3:548                3: 52      Median :3.000
##  4: 92                           Mean   :2.845
##                                  3rd Qu.:4.000
##                                  Max.   :4.000
##
##  Most.valuable.available.asset  Age..years.    Concurrent.Credits
##  1:282                         Min.   : 1.00   1:139
##  2:232                         1st Qu.: 9.00   2: 47
##  3:332                         Median :15.00   3:814
##  4:154                         Mean   :17.52
##                                3rd Qu.:24.00
##                                Max.   :53.00
##
##  Type.of.apartment No.of.Credits.at.this.Bank Occupation No.of.dependents
##  1:179             Min.   :1.000              1: 22      1:845
##  2:714             1st Qu.:1.000              2:200      2:155
##  3:107             Median :1.000              3:630
##                    Mean   :1.407              4:148
##                    3rd Qu.:2.000
##                    Max.   :4.000
##
##  Telephone Foreign.Worker
##  1:596     1:963
##  2:404     2: 37
##
##
##
##
##
```

Create Test and Training Datasets

To do so I created a function that splits the data into a 70/30 split and sets a seed. I then ran the bank dataset through my function.

```r
split_data <- function(x, p = 0.7, s = 777) {
  set.seed(s)
  index = sample(1:dim(x)[1])
  train = x[index[1:floor(dim(x)[1]*p)],]
  test = x[index[((ceiling(dim(x)[1]*p))+1):dim(x)[1]],]
  return(list(train = train, test = test))
}

bank_split <- split_data(bank)

bank_test <- bank_split$test

bank_train <- bank_split$train
```

Decision Tree Model

Build classification tree model using rpart.

```r
bank_rp <- rpart(Creditability~., data = bank_train)
```

Interpret Model

First, during the process of decision induction, we have to consider a statistic evaluation to partition the data into different partitions in accordance with the assessment result. Then, as we have determined by the child node, we can repeatedly perform the splitting until the stop criteria is satisfied.

Lets see how the model looks by using the printcp function to view the metrics of the complexity parameter. The CP serves as a penalty to control the size of the tree. In short, the greater the CP value, the fewer the number of splits there are.

- The output value represents the average deviance of the current tree divided by the average deviance of the null tree.
- A xerror value represents the relative error estimated by a 10-fold classification.
- xstd stands for the standard error of the relative error.

Taking a look at the output below, we see the xerror plateau at about 0.93 at the 10th split of the decision tree. Early on I find this indicative of overfitting being a problem with our model, because as we add more splits past 8 that we see less to no change in important error rates. This thought is somewhat validated in a root node error rate of 0.3.

```r
printcp(bank_rp)
```

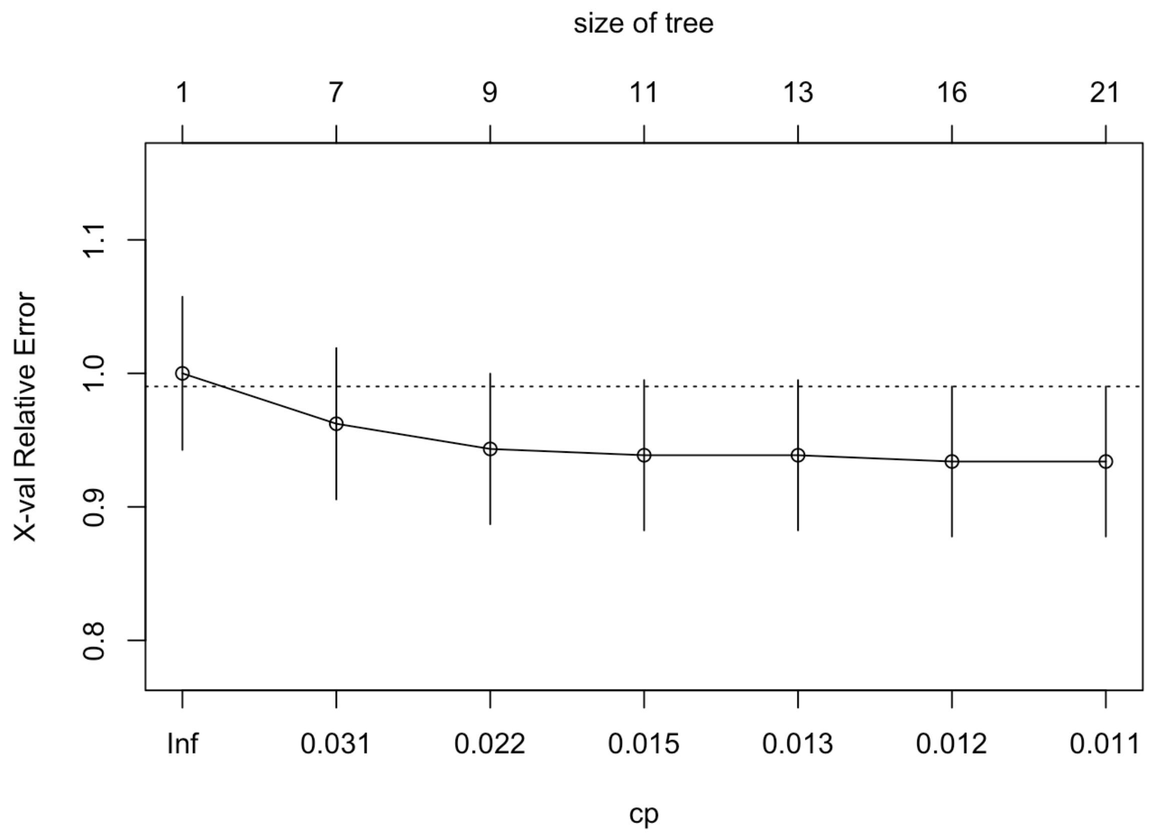
```
##
## Classification tree:
## rpart(formula = Creditability ~ ., data = bank_train)
##
## Variables actually used in tree construction:
##  [1] Account.Balance                   Age..years.
##  [3] Credit.Amount                     Duration.in.Current.address
##  [5] Duration.of.Credit..month.        Length.of.current.employment
##  [7] Most.valuable.available.asset     Occupation
##  [9] Payment.Status.of.Previous.Credit Purpose
## [11] Sex...Marital.Status              Value.Savings.Stocks
##
## Root node error: 212/700 = 0.30286
##
## n= 700
##
##         CP nsplit rel error  xerror     xstd
## 1 0.033019      0   1.00000 1.00000 0.057345
## 2 0.028302      6   0.75000 0.96226 0.056711
## 3 0.016509      8   0.69340 0.94340 0.056379
## 4 0.014151     10   0.66038 0.93868 0.056294
## 5 0.012579     12   0.63208 0.93868 0.056294
## 6 0.012264     15   0.59434 0.93396 0.056208
## 7 0.010000     20   0.53302 0.93396 0.056208
```

Lets now graphically view the complexity parameters. plotcp generates an information graphic of the CP table.

- x axis illistrates the cp value
- y axis illistrates the relative error
- upper x axis displays the size of the tree
- dotted line indicates the upper limit of the standard deviation

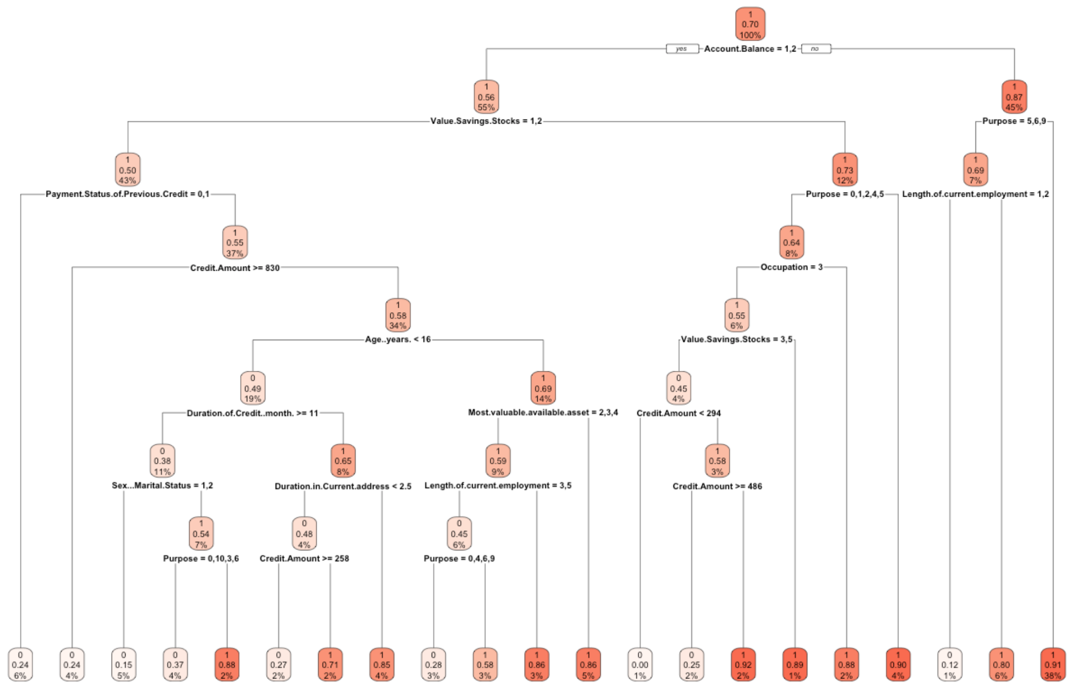
Looking at the map of the complexity parameter, 9 looks like the optimal split of our decision tree without overfitting.

```
plotcp(bank_rp)
```



Plotting our initial decision tree model to get a better view of splits and the metrics associated left us with some interesting output. We can see the variables ordered by importance and the subsequent splits of those variables. Nothing suprising here, but good that our common sense is statistically backed.

```
rpart.plot(bank_rp, box.palette = "Reds")
```



Predict Model 1

Use predict funtion to generate a classification table for the testing dataset.

```
predictions <- predict(bank_rp, bank_test, type = "class")

table(bank_test$Creditability, predictions)
```

```
##    predictions
##       0   1
##   0  38  50
##   1  41 171
```

We can see that the model is not performing very well, but does have some predictive power. This average predictability is affirmed by our accuracy score of 0.69 and our low postitive predictive value of 0.48.

Looking at the confusion matrix we continue to see the (what I think is) the overfitting of the training dataset in the predictive power of our decision tree model.

```
confusionMatrix(table(predictions, bank_test$Creditability))
```

```
## Confusion Matrix and Statistics
##
##
## predictions   0   1
##           0  38  41
##           1  50 171
##
##                Accuracy : 0.6967
##                  95% CI : (0.6412, 0.7482)
##     No Information Rate : 0.7067
##     P-Value [Acc > NIR] : 0.6739
##
##                   Kappa : 0.2458
##  Mcnemar's Test P-Value : 0.4017
##
##             Sensitivity : 0.4318
##             Specificity : 0.8066
##          Pos Pred Value : 0.4810
##          Neg Pred Value : 0.7738
##              Prevalence : 0.2933
##          Detection Rate : 0.1267
##    Detection Prevalence : 0.2633
##       Balanced Accuracy : 0.6192
##
##        'Positive' Class : 0
##
```

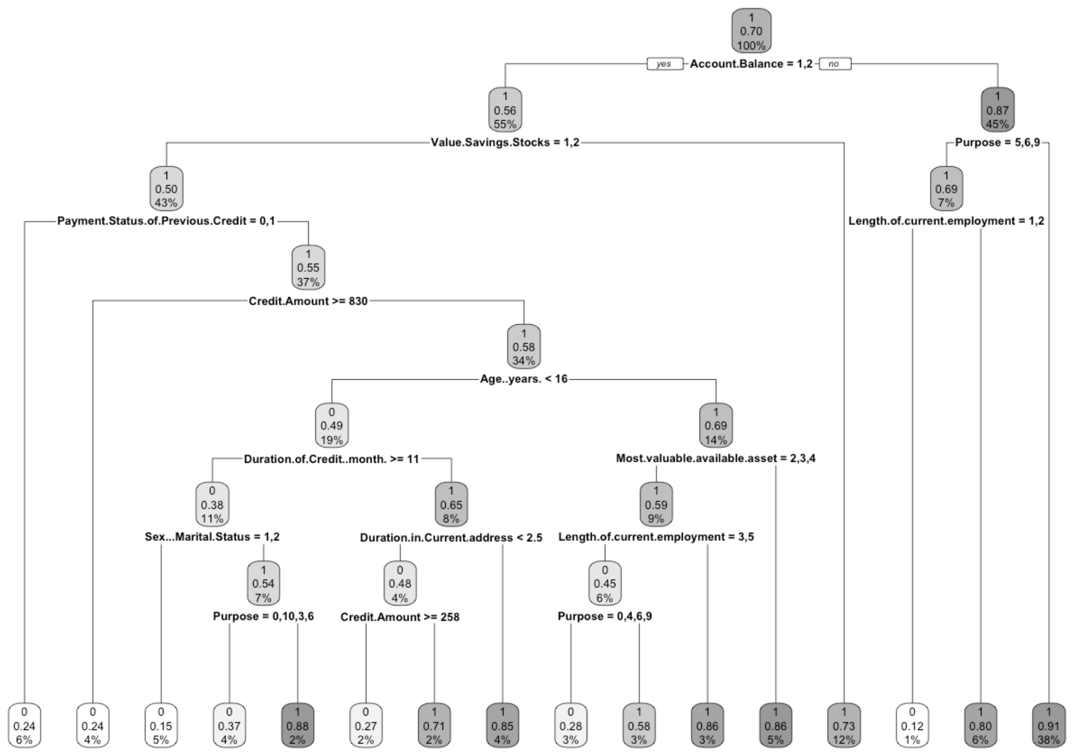
Prune, Interpret, Predict Model

I will prune the decision tree based on complexity parameter the optimal cp and xerror values.

```
bank_rp2 <- prune(bank_rp, cp = bank_rp$cptable[which.min(bank_rp$cptable[,"xerror"]), "CP"])
```

Let’s see what our newly pruned model looks like in a decision tree then test the new model on the test dataset.

```
rpart.plot(bank_rp2, box.palette = "Grays")
```



```
predictions2 <- predict(bank_rp2, bank_test, type = "class")
```

Now looking at our new confusion matrix with the pruned decision tree model, we see a mild uptick in accuracy and an increase in specificity. Our positive predictive value has stayed essentially the same. This does prove that we needed to prune the tree to make it more durable, however the predictive power of the bank_test dataset did not have any dramatic upticks.

```
confusionMatrix(table(predictions2, bank_test$Creditability))
```

```
## Confusion Matrix and Statistics
##
##
## predictions2   0   1
##            0  38  40
##            1  50 172
##
##                Accuracy : 0.7
##                  95% CI : (0.6447, 0.7513)
##     No Information Rate : 0.7067
##     P-Value [Acc > NIR] : 0.6273
##
##                   Kappa : 0.2515
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.4318
##             Specificity : 0.8113
##          Pos Pred Value : 0.4872
##          Neg Pred Value : 0.7748
##              Prevalence : 0.2933
##          Detection Rate : 0.1267
##    Detection Prevalence : 0.2600
##       Balanced Accuracy : 0.6216
##
##        'Positive' Class : 0
##
```

Random Forest Model

The next method we will use to classify the credibility of customers at the german bank will be the random forest method, which is essentially an ensemble model of decision trees combining the base principles of bagging with random feature selection to add additional diversity to the decision tree models.

After the ensemble of trees is generated, the model uses a vote to combine the trees' predictions. The ensemble uses only a small, random portion of the full feature set, random forests can handle extremely large datasets, and its error rates for most learning tasks are sufficient comparatively.

```
bank_rf <- randomForest(Creditability~., data = bank_train)
```

Interpret Model

We see the default 500 tree split tied to 4 variables at each split. Also note the out of bag error rate (unbiased estimate of the test set error), a number of 25% is not very good. Note the comparatively large type 1 (false positive) error in comparison to our type 2, false negative error. This means that our model might be prone to predicting a larger amount of people to be credit worthy, whom are actually not. Lets see if that holds up in our actual model.

```
bank_rf
```

```
##
## Call:
##  randomForest(formula = Creditability ~ ., data = bank_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24.29%
## Confusion matrix:
##    0   1 class.error
## 0 78 134  0.63207547
## 1 36 452  0.07377049
```

Predict Model and Conclusion

Now lets see how well we predict the test dataset. Whoa! Way better than I thought. Looks like we have properly predicted around 75% of the bank creditability numbers, with a decent p value, okay sensitivity and great specificity. This is an improvement in accuracy from both of the previous decision trees. Especially notice the larger decrease in type 1 errors. If I were this german bank, I would prioritize the reduction of these errors to type 2 errors to better protect the bank against large losses.

```
predictions3 <- predict(bank_rf, bank_test, type = "response")

confusionMatrix(table(predictions3, bank_test$Creditability))
```

```
## Confusion Matrix and Statistics
##
##
## predictions3   0   1
##            0  32  19
##            1  56 193
##
##                Accuracy : 0.75
##                  95% CI : (0.697, 0.798)
##     No Information Rate : 0.7067
##     P-Value [Acc > NIR] : 0.05483
##
##                   Kappa : 0.3124
##  Mcnemar's Test P-Value : 3.226e-05
##
##             Sensitivity : 0.3636
##             Specificity : 0.9104
##          Pos Pred Value : 0.6275
##          Neg Pred Value : 0.7751
##              Prevalence : 0.2933
##          Detection Rate : 0.1067
##    Detection Prevalence : 0.1700
##       Balanced Accuracy : 0.6370
##
##        'Positive' Class : 0
##
```

In the end I find that the random forest model is 5% more predictive than the pruned decision tree. In the setting of a bank loan I also think that putting someone’s information into a system and getting a more accurate creditability score, for the bank, will take priority over model that is less accurate with visability, such as the decision tree.