Lending Club - Trees and GLM models to Predict Funding

Nahel Rifai

4/18/2017

We used a data set of ~24,000 loan applications on Lending Club (wtih dimensions such as loan amount, annual income, home ownership status, etc). Based on each loan profile and credit report, every loan is assigned a "grade" by Lending Club. We create a new column on the data set called "High Grade". Highgrade takes 1 if the grade assigned is A or B. In this project, we create a classification tree and glm model that is trained on whether a loan will be assigned a high grade or not. Then, we run that model using test data in order to predict if a new loan will be assigned a good grade.

Setting up the working directory and libraries

```
setwd("~/Desktop/Prasanna/LendingClub")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rpart)
```

1. Data Loading and Cleanup

```
data = read.csv("LoanStats3c.csv", skip = 1, header = TRUE)
rows_number = (nrow(data)-2)
data = data[1:rows_number,]
```

2. Descriptive Statistics

Proportion of loans with "highgrade"

```
data$highgrade = (data$grade == "A" | data$grade == "B")
data$highgrade = as.integer(as.logical(data$highgrade))
highgrade_counter = data %>% group_by(highgrade) %>% tally()
proportion_highgrade =
  (highgrade_counter[2,2])/(highgrade_counter[1,2]+highgrade_counter[2,2])
proportion_highgrade[1,]
```

```
## [1] 0.4160905
```

T-test - Highgrade by annual income

Results: 0.463 probability of getting a "highgrade" loan if you're not "high income" 0.558 probability of getting a "highgrade" loan if you're "high income"

```r
median_annual_inc = median(data$annual_inc)
data$high_income = data$annual_inc >= median_annual_inc
t.test(data$high_income ~ data$highgrade)
```

```
##
##  Welch Two Sample t-test
##
## data:  data$high_income by data$highgrade
## t = -45.578, df = 211670, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.09883760 -0.09068745
## sample estimates:
## mean in group 0 mean in group 1
##       0.4632375       0.5580001
```

T-test - Highgrade by Loan Amount

Results: 0.536 probability of getting a "highgrade" loan if loan is "small" 0.469 probability of getting a "highgrade" loan if loan is "large"

```r
median_loan_amount = median(data$loan_amnt)
data$high_loan = data$loan_amnt >= median_loan_amount
t.test(data$high_loan ~ data$highgrade)
```

```
##
##  Welch Two Sample t-test
##
## data:  data$high_loan by data$highgrade
## t = 32.054, df = 211060, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.06275072 0.07092443
## sample estimates:
## mean in group 0 mean in group 1
##       0.5361011       0.4692635
```

T-test - Highgrade by Home Ownership

Results: 0.405 probability of getting a "highgrade" loan if person does not rent 0.375 probability of getting a "highgrade" loan if person does rent

```r
data$rent = data$home_ownership == "RENT"
t.test(data$rent ~ data$highgrade)
```

```
##
##  Welch Two Sample t-test
##
## data:  data$rent by data$highgrade
## t = 14.687, df = 212880, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.02591320 0.03389428
## sample estimates:
## mean in group 0 mean in group 1
##       0.4057898       0.3758861
```

3. Build a logistic classifier on the training data

```
##Creating the glm model
glm.model = glm(highgrade ~ annual_inc + home_ownership + loan_amnt,
                 data=data,
                 family = binomial(link="logit")
)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm.model)
```

```
##
## Call:
## glm(formula = highgrade ~ annual_inc + home_ownership + loan_amnt,
##     family = binomial(link = "logit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -1.0284  -0.8749   1.2716   1.7719
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            7.522e+00  2.666e+01   0.282    0.778
## annual_inc             7.281e-06  1.149e-07  63.376   <2e-16 ***
## home_ownershipMORTGAGE -7.699e+00  2.666e+01  -0.289    0.773
## home_ownershipOWN      -7.774e+00  2.666e+01  -0.292    0.771
## home_ownershipRENT     -7.853e+00  2.666e+01  -0.294    0.768
## loan_amnt              -4.331e-05  5.959e-07 -72.671   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 319984  on 235628  degrees of freedom
## Residual deviance: 312626  on 235623  degrees of freedom
## AIC: 312638
##
## Number of Fisher Scoring iterations: 6
```

The predicting effectiveness of the model on the training data. It has a 0.594 probability of being accurate.

```
prediction_model = predict(glm.model)
predict.df = cbind((prediction_model[] > 0),data)
##If coefficient higher than 0, then predict TRUE
colnames(predict.df)[1] = "Prediction"
effectiveness.df = cbind(predict.df$Prediction, data$highgrade)
effectiveness.df = as.data.frame(effectiveness.df)
colnames(effectiveness.df)[1] = "Prediction"
colnames(effectiveness.df)[2] = "Actual"
mean(effectiveness.df$Prediction == effectiveness.df$Actual)
```

```
## [1] 0.59474
```

Benchmarking against random 1s and 0s

```
data$random <- sample(0:1, size = nrow(data), replace = TRUE)
benchmark_one_df = cbind(data$random, data$highgrade)
benchmark_one_df = as.data.frame(benchmark_one_df)
```

```r
mean(benchmark_one_df$V1 == benchmark_one_df$V2)
```

```
## [1] 0.5002016
```

Benchmarking against all 0s

```r
data$zeros <- sample(0, size = nrow(data), replace = TRUE)
benchmark_two_df = cbind(data$zeros, data$highgrade)
benchmark_two_df = as.data.frame(benchmark_two_df)
mean(benchmark_two_df$V1 == benchmark_two_df$V2)
```

```
## [1] 0.5839095
```

4. Supervised Learning

Building the classification tree

```r
fit = rpart(highgrade ~ annual_inc + loan_amnt + home_ownership,
            data = data, method = "class")
fit
```

```
## n= 235629
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 235629 98043 0 (0.5839095 0.4160905)
##    2) loan_amnt>=28012.5 20518  4857 0 (0.7632810 0.2367190) *
##    3) loan_amnt< 28012.5 215111 93186 0 (0.5668004 0.4331996)
##      6) annual_inc< 72122.5 139110 53170 0 (0.6177845 0.3822155) *
##      7) annual_inc>=72122.5 76001 35985 1 (0.4734806 0.5265194)
##       14) annual_inc< 99853.58 41701 20431 0 (0.5100597 0.4899403)
##         28) loan_amnt>=20025 9771  3903 0 (0.6005527 0.3994473) *
##         29) loan_amnt< 20025 31930 15402 1 (0.4823677 0.5176323) *
##       15) annual_inc>=99853.58 34300 14715 1 (0.4290087 0.5709913) *
```

Predicting Effectiveness of the Classification Tree on the training data. Probability of 0.51 of predicting correct outcome. It is a better predictor than the logistic regression.

```r
z = predict(fit, type="class")
data$tree_prediction = z
tree_prediction_df = NULL
tree_prediction_df$Prediction = data$tree_prediction
tree_prediction_df$Actual = data$highgrade
tree_prediction_df = as.data.frame(tree_prediction_df)
mean(tree_prediction_df$Prediction == tree_prediction_df$Actual)
```

```
## [1] 0.6093562
```

5. Model Performance on the Test Data

Loading and cleaning up the Test Data

```r
test_data = read.csv("LoanStats3d.csv", skip = 1, header = TRUE)
rows_number = (nrow(test_data)-2)
test_data = test_data[1:rows_number,]
test_data$highgrade = (test_data$grade == "A" | test_data$grade == "B")
```

Predicting highgrade on test data with the GLM model. Effectiveness of 0.58. Almost the same as on training data.

```
##glm model
glm_prediction_results = predict(glm.model,test_data)
test_data$GLM_Prediction = (glm_prediction_results > 0) #setting coefficient threshold
mean(test_data$highgrade == test_data$GLM_Prediction)
```

```
## [1] 0.5771311
```

Predicting highgrade on test data with Tree Classification Model. Effectiveness of 0.60, very close to prediction on training data.

```
tree_prediction_results <- predict(fit, test_data)
test_data$Tree_Prediction = (tree_prediction_results[,2] > tree_prediction_results[,1])
mean(test_data$highgrade == test_data$Tree_Prediction)
```

```
## [1] 0.6053076
```

Benchmarking against random 1s and 0s on Test Data.

```
test_data$random <- sample(0:1, size = nrow(test_data), replace = TRUE)
##assigning random 1s and 0s
final_benchmark_one_df = cbind(test_data$random, test_data$highgrade)
final_benchmark_one_df = as.data.frame(final_benchmark_one_df)
mean(final_benchmark_one_df$V1 == final_benchmark_one_df$V2)
```

```
## [1] 0.5003669
```

Benchmarking against all 0s on Test Data.

```
test_data$zeros <- sample(0, size = nrow(test_data), replace = TRUE)
##assigning random 1s and 0s
final_benchmark_two_df = cbind(test_data$zeros, test_data$highgrade)
final_benchmark_two_df = as.data.frame(final_benchmark_two_df)
mean(final_benchmark_two_df$V1 == final_benchmark_two_df$V2)
```

```
## [1] 0.5465584
```