The Big Mac

Adam Starr, Neel Adhiraj Kumar, Tim Schaaff

12/7/2016

```
## Warning: package 'glmnet' was built under R version 3.1.3

## Loading required package: Matrix
## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.1.3

## Loaded glmnet 2.0-3

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

For the *Give Me Some Credit* competition on Kaggle, our goal is to predict the probability a given borrower will default on a loan, given information about the borrower's financial circumstances and credit history. We called our initial model *The Big Mac* since it was a sort of model within a model (like the Big Mac's bun inside of a bun). As it turns out, much as the Big Mac burger is a bit of a gimmick, so too did our original model perform poorly. Ultimately, our best performing model was a random forest trained on a weighted subset of the training data.

Imputation and Feature Extraction

We are given training data with 150,000 observations of the following categories:

- Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits
- Age of borrower in years
- Number of times borrower has been 30-59 days past due but no worse in the last 2 years.
- Monthly debt payments, alimony,living costs divided by monthly gross income
- Monthly income
- Number of open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)
- Number of times borrower has been 90 days or more past due.
- Number of mortgage and real estate loans including home equity lines of credit
- Number of times borrower has been 60-89 days past due but no worse in the last 2 years.
- Number of dependents in family excluding themselves (spouse, children etc.)

Our target variable we wish to predict is whether the borrower will experience a serious delinquency within 2 years. The most important step in developing this model is probably the imputation. Our given data is messy, to say the least, with missing values in the monthly income and number of dependents categories, and numerous outliers.

```
summary(cs.training$MonthlyIncome)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0    3400    5400    6670    8249 3009000   29731
```

```
summary(cs.training$NumberOfDependents)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   0.000   0.757   1.000  20.000    3924
```

```
summary(cs.training$DebtRatio)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      0.0      0.2      0.4    353.0      0.9 329700.0
```

Before we impute missing values, we observe extreme outliers in the debt ratio category. Since debt ratio is calculated as the monthly debt payments divided by monthly income, debt ratios in the hundreds and thousands do not pass the common sense test. When we dig deeper, we notice that data entries with 0 or NA income almost exclusively have these outlier debt ratios.

```
summary(cs.training$DebtRatio[which(cs.training$MonthlyIncome == 0
                              | is.na(cs.training$MonthlyIncome))])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     122    1144    1668    2373  329700
```

Technically, debt ratio is undefined for a borrower with 0 or NA income. We infer that debt ratio is calculated for these borrowers by dividing monthly debt payments by 1 (that is, debt ratio is equal to monthly debt payments). With this assumption, we create an extra variable for monthly debt payments. If monthly income is 0 or NA, we set this equal to the debt ratio. Otherwise, we calculate it as (monthly income)*(debt ratio). We proceed with imputation by leaving out debt ratio as a predictor in regression, and use monthly debt payments in its place.

We impute missing income and dependents values by iteratively running regressions. We begin by replacing missing dependents values with the mean number of dependents. We then predict the missing incomes by running an ordinary linear regression on the data with non-missing incomes. Then we update the missing dependents values using a Poisson regression. We iterate over this regression process 10 times, a sufficient number to reach approximate convergence in the predicted values.

Once the missing values have been imputed, we back into the debt ratio for those data entries with 0 or NA income. If income was NA, we use the newly imputed income to infer a debt ratio. If income is 0, we set debt ratio equal to monthly debt payments (admittedly, this is not ideal since the debt ratio is still undefined in these cases).

In addition to imputation, we attempt to infer additional relevant features from the data. This can be particularly useful for those categories which have outliers, but for which we may not be comfortable adjusting data entries for. For example, it does not make sense for revolving utilization of credit lines to be greater than 1, and yet there are extreme outliers in this category.

```
summary(cs.training$RevolvingUtilizationOfUnsecuredLines)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     0.03     0.15     6.05     0.56 50710.00
```

Unlike with the debt ratio, we may not be comfortable altering these data entries to make them more reasonable. In that case, we can create an indicator variable to classify data entries with outlier values. In total, we created the following indicator variables:

- If the debt ratio is greater than 1
- If the debt ratio is greater than 100
- If the total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits is greater than 1
- If the total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits is greater than 100
- If monthly income is below 100
- If monthly income is zero
- If there are more than 20 credit lines
- If there are more than 4 real estate loans

In addition, we added log transformations of the following:

- The total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits
- Monthly income

Here is the imputation and feature extraction code:

```
process.data <- function(x) {

  # NAs in cols 7 (mthly income) and 12 (# dependents)
  income.nas <- is.na(x$MonthlyIncome)
  dep.nas <- is.na(x$NumberOfDependents)
  zeroincome <- x$MonthlyIncome == 0 & is.na(x$MonthlyIncome) == FALSE

  # we have absurd values for debt ratio, since we have NAs for
  # monthly income and debt ratio is calculated from it
  # to fix this, we define a new variable, monthly debt payments
  # calculate monthly debt payments using debt ratio and monthly income
  # if income is NA or 0, monthly debt payments is equal to debt ratio
  x$MonthlyDebtPayments[income.nas | zeroincome] <- x$DebtRatio[income.nas | zeroincome]
  x$MonthlyDebtPayments[!income.nas & !zeroincome] <- x$MonthlyIncome[!income.nas & !zeroincome] *
    x$DebtRatio[!income.nas & !zeroincome]

  # mean # of dependents, removing NAs
  dep.guess <- mean(x$NumberOfDependents, na.rm = TRUE)

  # replace dependents NAs with guess
  x$NumberOfDependents[dep.nas] <- dep.guess

  n <- 10 # number of iterations to run

  for (i in 1:n) {
```

```
    # fit regression to data without missing income, use to predict missing values
    income.lm <- lm(MonthlyIncome ~ RevolvingUtilizationOfUnsecuredLines + age +
                      NumberOfOpenCreditLinesAndLoans + NumberRealEstateLoansOrLines +
                      NumberOfDependents + NumberOfTime30.59DaysPastDueNotWorse +
                      NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
                      MonthlyDebtPayments,
                    data = x[!income.nas,])
    x$MonthlyIncome[income.nas] <- predict(income.lm, x[income.nas,])

    # fit poisson regression to data without # of dependents, use to predict missing values
    dep.lm <- glm(NumberOfDependents ~ RevolvingUtilizationOfUnsecuredLines + age +
                    NumberOfOpenCreditLinesAndLoans + NumberRealEstateLoansOrLines +
                    MonthlyIncome + NumberOfTime30.59DaysPastDueNotWorse +
                    NumberOfTimes90DaysLate + NumberOfTime60.89DaysPastDueNotWorse +
                    MonthlyDebtPayments,
                  data = x[!dep.nas,], family = poisson())
    x$NumberOfDependents[dep.nas] <- predict(dep.lm, x[dep.nas,], type = 'response')
  }

  #imputed data
  summary(x$MonthlyIncome)
  summary(x$NumberOfDependents)

  # back into DebtRatio for imputed incomes
  x$DebtRatio[income.nas] <- x$MonthlyDebtPayments[income.nas]/x$MonthlyIncome[income.nas]
  summary(x$DebtRatio)

  # extract features from data
  x$DebtRatioAbove1 <- as.integer(x$DebtRatio > 1)
  x$DebtRatioAbove100 <- as.integer(x$DebtRatio > 100)
  x$LogRevolUtil <- log(x$RevolvingUtilizationOfUnsecuredLines + 1)
  x$RevolUtilAbove1 <- as.integer(x$RevolvingUtilizationOfUnsecuredLines > 1)
  x$RevolUtilAbove100 <- as.integer(x$RevolvingUtilizationOfUnsecuredLines > 100)
  x$ZeroIncome <- as.integer(x$MonthlyIncome == 0)
  x$IncomeBelow100 <- as.integer(x$MonthlyIncome < 100)
  x$LogIncome <- log(x$MonthlyIncome + 1)
  x$CreditLinesAbove20 <- as.integer(x$NumberOfOpenCreditLinesAndLoans > 20)
  x$RealEstateLoansAbove4 <- as.integer(x$NumberRealEstateLoansOrLines > 4)

  return(x)
}

cs.imp <- process.data(cs.imp)
```

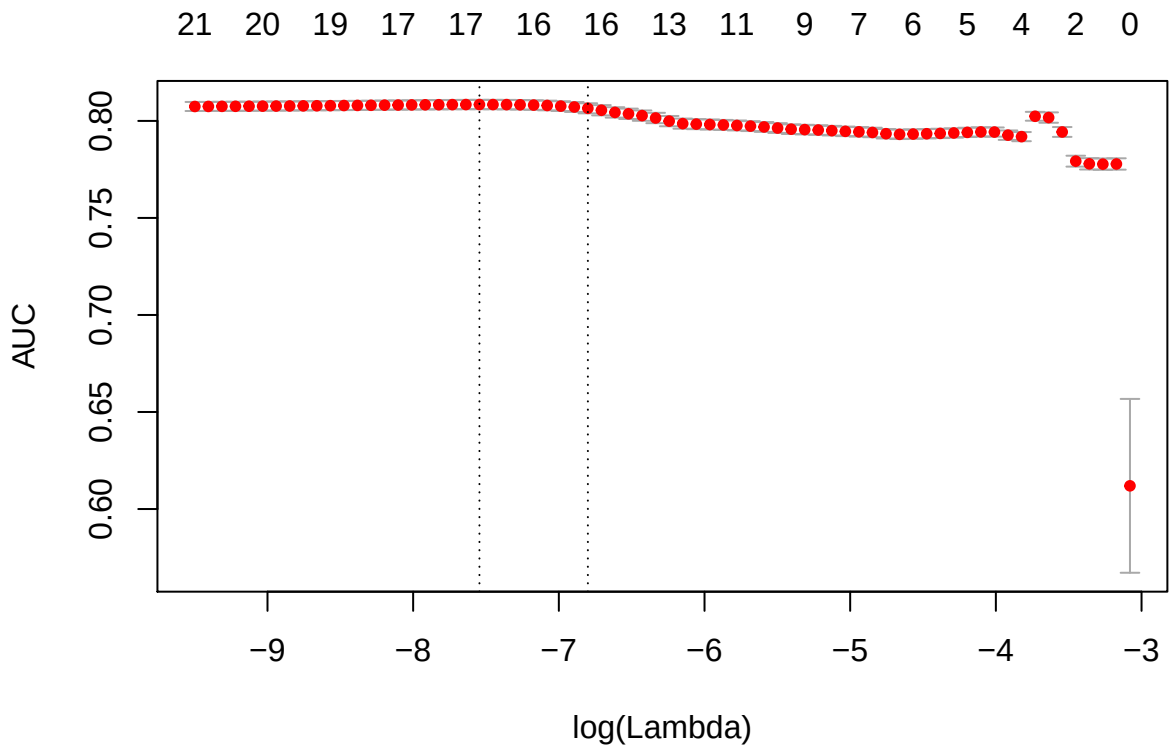
Logistic Regression

One approach was logistic regression with LASSO penalty. LASSO was used to generate a sparse solution. We used cross validation on AUC to pick our lambda value.

```
cvfit <- cv.glmnet(as.matrix(cs.imp[,2:22]),
                   as.matrix(cs.imp[,1]), family = "binomial",
                   type.measure = "auc", parallel = TRUE)
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
plot(cvfit)
```




```
cvfit$lambda.min #the best lambda
```

```
## [1] 0.0005287667
```

```
coef(cvfit, s = "lambda.min") #the coefients for our variables
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                                                1
## (Intercept)                         -2.806600e+00
## RevolvingUtilizationOfUnsecuredLines -7.086538e-04
## age                                 -1.499843e-02
## NumberOfTime30.59DaysPastDueNotWorse  2.440915e-01
## DebtRatio                           1.143227e-07
## MonthlyIncome                       -1.448238e-05
## NumberOfOpenCreditLinesAndLoans      2.229340e-02
## NumberOfTimes90DaysLate              1.769120e-01
## NumberRealEstateLoansOrLines         .
## NumberOfTime60.89DaysPastDueNotWorse -3.976893e-01
## NumberOfDependents                   6.939877e-02
## MonthlyDebtPayments                  3.506181e-06
## DebtRatioAbove1                      3.262123e-01
## DebtRatioAbove100                    .
## LogRevolUtil                         3.243034e+00
## RevolUtilAbove1                      6.295685e-01
```

```
## RevolUtilAbove100                    -2.257367e+01
## ZeroIncome                            .
## IncomeBelow100                       -1.427933e+00
## LogIncome                            -6.613972e-02
## CreditLinesAbove20                    .
## RealEstateLoansAbove4                 9.753228e-01
```

Clearly, LASSO lead to a few variables being dropped.

After imputation, our regression led to an AUC of 0.808178 on the test data.

Random Forests

We also used random forests to build a model.

```
forest <- tuneRF(x=cs.imp[,2:23],y = as.factor(SeriousDlqin2yrs),
                 stepFactor = 2, improve=0.05, doBest=TRUE, ntree=501)
```

Using the stock implementation with the entire training set, we had an AUC of 0.8521 on the test set. This is not bad, considering that the highest performing models in the Kaggle competition were combinations of different learning ensembles. But we still think we can do better.

```
sum(cs.training$SeriousDlqin2yrs / dim(cs.training)[1])
```

```
## [1] 0.06684
```

We notice that cases of defaults are quite rare, representing only 6.7% of the training data. If we wish to train our learners to identify defaults, we may consider taking a weighted subsample of the training data, where defaults are over-represented. So we trained a random forest on a 20,000 element subsample, with defaults weighted twice as heavily. This forest yielded an AUC of 0.85529 on the test data.

This leads us to *The Big Mac*. We may reasonably guess that individual models will predict with greater accuracy on certain types of data, while performing poorly with others. Based on this idea, we may try to combine the predictions of our individual models, in the hope that the combination of their predictions will be more accurate with cases that are difficult to identify. This is the bun inside the bun, so to speak. So we trained a random forest on a combination of the data, the logistic regression predictions, the stock random forest implementation, and the weighted random forest implementation. Sadly, this forest gave an AUC of 0.751441. Our grand idea actually under-performed all of its constituent models. So we went back to the drawing board.

We returned to training random forests on weighted subsamples of the training data, since this produced good results for with the Big Mac. In total, we trained random forests on 20,000 element subsamples with defaults weighted 4x, 8x, 32x, 64x, and 128x more heavily than non-defaults. This gave us the following AUC results on the test data.

Here are tables of the importance of each variable in the different weighted forests. We are considering mean decrease in in accuracy as our measure of importance.

```
importance(forest2)
```

```
##                                         0         1
## RevolvingUtilizationOfUnsecuredLines 57.988294 -15.925870
## age                                  27.950082  21.013501
```

```
## NumberOfTime30.59DaysPastDueNotWorse 44.677504  61.965770
## DebtRatio                            58.319379  -8.120812
## MonthlyIncome                        53.816641 -32.212988
## NumberOfOpenCreditLinesAndLoans      48.365659 -11.920441
## NumberOfTimes90DaysLate              90.659076 106.567981
## NumberRealEstateLoansOrLines         28.036641  -5.306793
## NumberOfTime60.89DaysPastDueNotWorse 72.612407  73.689871
## NumberOfDependents                   17.432077   3.324398
## MonthlyDebtPayments                  58.991943 -20.902271
## DebtRatioAbove1                      13.922026  -7.622136
## DebtRatioAbove100                    14.092278  -1.874334
## LogRevolUtil                         56.691777 -14.735300
## RevolUtilAbove1                       2.313851  19.242509
## RevolUtilAbove100                     9.264783  -4.775778
## ZeroIncome                            3.320152  -1.936629
## IncomeBelow100                       12.947542   1.336644
## LogIncome                            51.177868 -31.759406
## CreditLinesAbove20                    8.724313  -2.998381
## RealEstateLoansAbove4                 1.966852   7.881900
##                                      MeanDecreaseAccuracy MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines            59.340637       427.719919
## age                                             37.399669       290.178082
## NumberOfTime30.59DaysPastDueNotWorse            72.016628       233.262631
## DebtRatio                                       59.410471       319.761348
## MonthlyIncome                                   53.123294       293.760352
## NumberOfOpenCreditLinesAndLoans                 45.341283       216.639158
## NumberOfTimes90DaysLate                        124.150161       381.681391
## NumberRealEstateLoansOrLines                    28.262375        89.568300
## NumberOfTime60.89DaysPastDueNotWorse            97.887441       232.556394
## NumberOfDependents                              17.577382       115.809454
## MonthlyDebtPayments                             58.451025       318.956983
## DebtRatioAbove1                                 13.737078        12.580759
## DebtRatioAbove100                               14.319986         2.230451
## LogRevolUtil                                    58.121895       430.341229
## RevolUtilAbove1                                 12.695098        37.706532
## RevolUtilAbove100                                8.193037         1.226355
## ZeroIncome                                       2.801229         1.708953
## IncomeBelow100                                  13.463296         2.943999
## LogIncome                                       50.619244       295.665967
## CreditLinesAbove20                               7.712641         8.696928
## RealEstateLoansAbove4                            4.756705         8.037023
```

```
importance(forest4)
```

```
##                                      MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines      1323.100259
## age                                        811.676679
## NumberOfTime30.59DaysPastDueNotWorse       769.156665
## DebtRatio                                  913.382792
## MonthlyIncome                              823.783330
## NumberOfOpenCreditLinesAndLoans            602.359531
## NumberOfTimes90DaysLate                   1273.910865
## NumberRealEstateLoansOrLines               235.319114
## NumberOfTime60.89DaysPastDueNotWorse       576.941369
```

```
## NumberOfDependents                    317.754373
## MonthlyDebtPayments                   898.736516
## DebtRatioAbove1                        33.851117
## DebtRatioAbove100                       4.897891
## LogRevolUtil                         1323.589157
## RevolUtilAbove1                       123.957068
## RevolUtilAbove100                       4.904089
## ZeroIncome                              5.043744
## IncomeBelow100                          6.479067
## LogIncome                             828.735552
## CreditLinesAbove20                     24.176201
## RealEstateLoansAbove4                  20.420752
```

```
importance(forest8)
```

```
##                                     MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines      1339.132232
## age                                        819.272000
## NumberOfTime30.59DaysPastDueNotWorse       765.620605
## DebtRatio                                  907.428092
## MonthlyIncome                              827.560221
## NumberOfOpenCreditLinesAndLoans            611.626033
## NumberOfTimes90DaysLate                   1253.712147
## NumberRealEstateLoansOrLines               239.062371
## NumberOfTime60.89DaysPastDueNotWorse       617.506528
## NumberOfDependents                         309.037476
## MonthlyDebtPayments                        892.073760
## DebtRatioAbove1                             34.736339
## DebtRatioAbove100                            4.657974
## LogRevolUtil                              1310.294358
## RevolUtilAbove1                            124.740061
## RevolUtilAbove100                            5.093205
## ZeroIncome                                   4.972075
## IncomeBelow100                               6.175088
## LogIncome                                  830.986388
## CreditLinesAbove20                          24.458823
## RealEstateLoansAbove4                       19.418175
```

```
importance(forest32)
```

```
##                                     MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines      1720.289903
## age                                        985.640450
## NumberOfTime30.59DaysPastDueNotWorse      1065.882029
## DebtRatio                                 1072.979862
## MonthlyIncome                              971.609393
## NumberOfOpenCreditLinesAndLoans            729.897533
## NumberOfTimes90DaysLate                   1521.851243
## NumberRealEstateLoansOrLines               279.873696
## NumberOfTime60.89DaysPastDueNotWorse       787.801989
## NumberOfDependents                         367.471364
## MonthlyDebtPayments                       1054.696034
## DebtRatioAbove1                             40.439644
```

```
## DebtRatioAbove100                        5.328640
## LogRevolUtil                          1751.352543
## RevolUtilAbove1                        165.019632
## RevolUtilAbove100                        4.267873
## ZeroIncome                               6.158271
## IncomeBelow100                           7.912098
## LogIncome                              963.208332
## CreditLinesAbove20                      30.125280
## RealEstateLoansAbove4                   24.649379
```

```
importance(forest64)
```

```
##                                      MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines      1736.860287
## age                                        982.710163
## NumberOfTime30.59DaysPastDueNotWorse      1028.840091
## DebtRatio                                 1083.753857
## MonthlyIncome                              969.348306
## NumberOfOpenCreditLinesAndLoans            726.878303
## NumberOfTimes90DaysLate                   1552.508311
## NumberRealEstateLoansOrLines               275.564438
## NumberOfTime60.89DaysPastDueNotWorse       784.275213
## NumberOfDependents                         366.604848
## MonthlyDebtPayments                       1068.785097
## DebtRatioAbove1                             40.090107
## DebtRatioAbove100                            5.877870
## LogRevolUtil                              1751.871065
## RevolUtilAbove1                            159.413405
## RevolUtilAbove100                            5.416770
## ZeroIncome                                   5.836786
## IncomeBelow100                               8.174147
## LogIncome                                  967.495806
## CreditLinesAbove20                          28.765513
## RealEstateLoansAbove4                       23.710519
```

```
importance(forest128)
```

```
##                                      MeanDecreaseGini
## RevolvingUtilizationOfUnsecuredLines      1754.084246
## age                                        977.188582
## NumberOfTime30.59DaysPastDueNotWorse      1018.938063
## DebtRatio                                 1077.572058
## MonthlyIncome                              969.364414
## NumberOfOpenCreditLinesAndLoans            718.333700
## NumberOfTimes90DaysLate                   1631.979186
## NumberRealEstateLoansOrLines               276.183907
## NumberOfTime60.89DaysPastDueNotWorse       792.546916
## NumberOfDependents                         362.492170
## MonthlyDebtPayments                       1055.540824
## DebtRatioAbove1                             39.141239
## DebtRatioAbove100                            5.791686
## LogRevolUtil                              1687.738503
## RevolUtilAbove1                            159.995715
```

```
## RevolUtilAbove100                  5.965636
## ZeroIncome                         6.237231
## IncomeBelow100                     7.671059
## LogIncome                        966.376813
## CreditLinesAbove20                29.608018
## RealEstateLoansAbove4             23.300860
```

Default Subsample Weight	AUC
2x	0.8553
4x	0.8589
8x	0.8599
32x	0.8611
64x	0.8612
128x	0.8601

As one would expect, we eventually hit a point of diminishing returns. Ultimately, our best random forest model was trained on the subsample with defaults weighted 64x more heavily.