CREDIT SCORE MODELLING

ANDREW KINSMAN

MSc Data Analytics, Programming for Analytics 2015/16

Contents

Introduction

This project tackles a credit scoring problem that was the subject of an online competition run by Kaggle in 2011 [1]. Credit scoring is a standard procedure used by lenders to evaluate the potential risks of a loan, with a view to avoiding or reducing potential losses that would arise from bad debts [2]. Each client is given a credit score based on available information about that individual. Accurate credit scoring enables the lender to identify in advance which loans might be particularly high risk, so that they can perhaps act to prevent delinquency and/or make adequate provision for losses.

The task is to create a predictive model to determine whether or not an existing loan is a bad credit risk. Clearly there are only two possible outcomes: either the loan will become seriously delinquent (within a specified time period, in this case two years) or it won't. In this problem (known as a binary classification problem), the credit score of each loan represents the probability of it going bad. A credit score of close to 1 indicates a high credit risk, whereas a score close to zero is low risk.

The underlying idea of the model is to use data about loans for which the delinquency history is known to predict the credit risk of other loans, both current and future.

What are the Data?

Kaggle provide two sets of data: a training set for use in model creation and a test set which was used for assessing the competition entries. The training data contain 150,000 observations with 10 numerical explanatory variables or "predictors" (such as age, income, number of dependents, debt ratio). The test set has a further 101,503 observations with the same explanatory variables but without any values for the target variable, for which predictions need to be created. This target (or "dependent") variable is whether or not the borrower experiences serious delinquency within the next two years. Kaggle also supply a data dictionary explaining each variable in the data [3].

Exploratory Data Analysis

Before starting any data preparation, it is advisable to undertake some exploratory data analysis (EDA), looking at the underlying structure of the dataset and performing summaries and visualisations. This will enable better understanding of the data and perhaps highlight some issues that may require special attention. Are there any missing or clearly incorrect values? How is each variable distributed (wide or narrow range, normally or skewed, with many outliers or not etc.)? Do the variables have the correct type (character, factor, integer, date)? Etc. Furthermore, EDA may reveal insights into possible relationships within the data and help identify which approach should be taken to the modelling process.

First the target variable is considered. Only around 10,000 (6.7%) of the 150,000 training set loans resulted in serious delinquency *(fig.1)*. This severe imbalance between the two classes of the target variable could present problems when building a model, because the lack of minority class ("seriously delinquent") data may

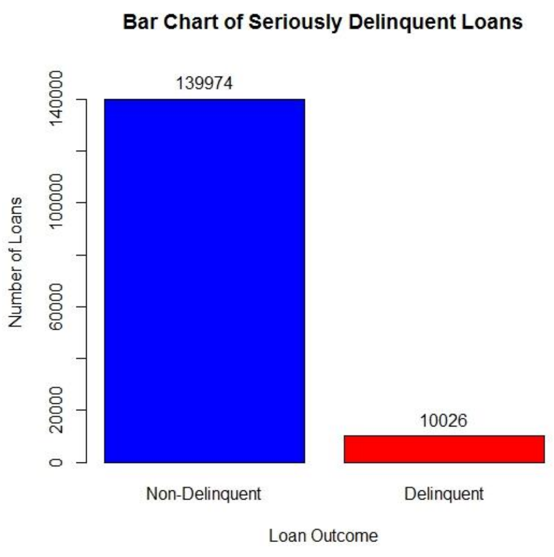

Figure 1

result in bias towards the majority classifier ("not seriously delinquent"). Such bias would cause over-optimistic estimates of the accuracy of the model, which would then fail to perform as well as expected when deployed on new data. Some methods for dealing with this imbalance will be discussed later in this report.

Further exploratory data analysis enables some initial observations to be made regarding the ten predictors, all of which (apart from one) are clearly skewed right:

1. The age variable has one erroneous value of zero, and might seem more or less normally distributed judging by the roughly bell-shaped histogram and fairly straight, diagonal middle part of the line in the Q-Q plot *(fig.2)*. However, judging normality based on histograms is dangerous because any change in the number of bins will affect its shape. Indeed, a Lilliefors (Kolmogorov-Smirnov) test rejects the null hypothesis that age is normally distributed.

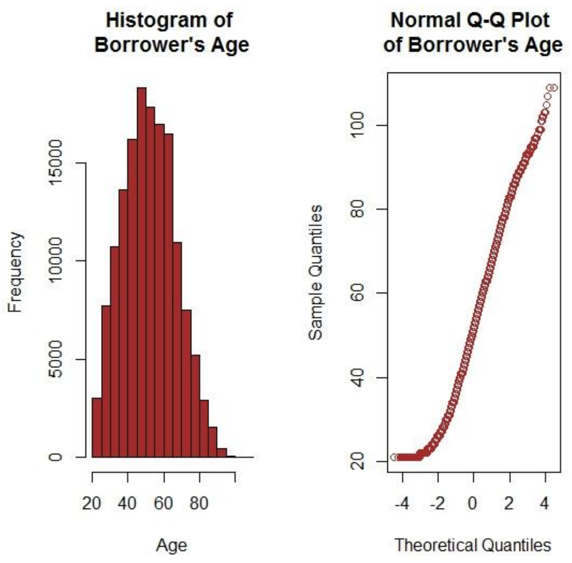

Figure 2

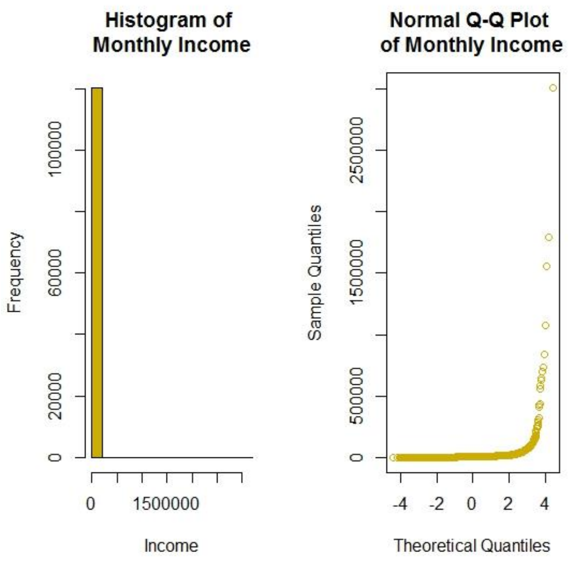

Figure 3

2. Unlike the age variable, monthly income has a heavily right-skewed histogram and no diagonal line in the Q-Q plot *(fig.3)*. The 104 borrowers earning more than 1MM per annum (the currency is unspecified) help to squash the monthly income default R boxplot almost flat, so in order to produce a standard-looking boxplot the outliers must be removed *(fig. 4)*. The summary statistics reveal that 50% of clients earn between 3400 and 8249 per month (third quartile minus first quartile) and also that nearly 20% of the monthly income values are missing, which is clearly something that will require attention.

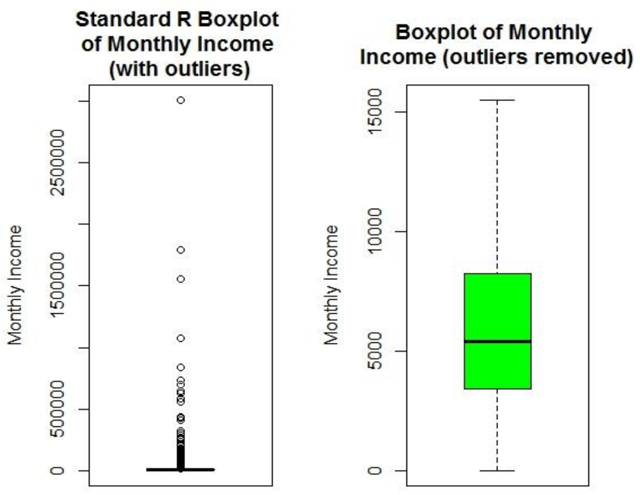

Figure 4

3. Most clients have between two and ten open credit lines and loans, but some have as many as 58 and there is clearly a right-skewed distribution.
4. The typical borrower has up to four dependents, but one has 13 and another has 20. There are nearly

4,000 (2.6%) missing values that will need to be addressed.

5. Clients typically have either two, one or zero real estate loans or lines, although one outlier borrower has as many as 54.
6. The third quartile of revolving utilisation of unsecured lines is just 0.56, but there are also some borrowers who possess hundreds or thousands of lines, going all the way up to over 50,000. This is one of several areas in which expert knowledge of credit scoring would be of assistance: How should these outliers be dealt with? Are they potentially important data points or simply erroneous data?
7. The third quartile of debt ratio is just 0.9, but there are also some clients with a debt ratio into the thousands, going all the way up to over 300,000. Again expert interpretation would be helpful here.
8. The other three variables, number of times 30-59 days, 60-89 days and 90+ days past due (late), each have the exact same count of outliers at values of 96 and 98. Both of these codes are commonly utilised in this kind of context to represent missing or corrupt data, coding issues etc.

Data Preparation

In this project the first data preparation step is to merge the training and test data together, so that dealing with any incorrect or missing values can be undertaken quickly and consistently on all data at the same time. There are various ways in which this cleaning can be carried out, including:

1. Rows with incorrect/missing values can be eliminated from the dataset.
2. Columns with incorrect/missing values can be removed from the dataset.
3. Some measurement of central location (e.g. the mean or median) can be substituted for the incorrect/missing values. Typically the median would be used as it is a more robust indicator of the central location than the mean, which is subject to influence by any extreme outliers.
4. A multiple imputation method can be used.

Rather than delete what might be useful data, an imputation approach was implemented. A simple approach of substituting the median was used for variables in which there were only a small amount of observations affected (replacing the “0” in the age column, and the “96”s and “98”s in each of the three columns relating to days late). However, while this approach is easy to implement, it may produce biased results in cases where the data is not missing completely at random. When there are larger numbers of missing or incorrect values (as with the monthly income and number of dependents variables), a multiple imputation approach is certainly preferable.

Multiple imputation works by generating predictions for each variable with missing values by utilising every other variable observation. It then uses those predictions to create plausible values for the missing data, with the process iterating until there is convergence over the missing values [4]. Several packages in R can perform multiple imputation, including the mice package that was used here to impute the missing values for monthly income and number of dependents.

Now that data preparation is complete, the data can be split back into the original training and test sets so that the model can be built on the training data and later assessed on the test data.

Having replaced the missing values, it is possible to construct a correlation matrix containing every variable in the training data. The strongest correlation (+0.43) is between the numbers of open credit and real estate loans or lines. The number of times past due for 30-59 day, 60-89 days and 90+ days are also fairly well correlated not only with the target variable, but also with each other. These correlations between predictors indicate that there could possibly be a multicollinearity issue, whereby it would be problematic to distinguish between the individual effects of different variables.

Analysing these correlations, a boxplot of the number of times 30-59 days past due suggests that (not surprisingly) there is a much higher likelihood of delinquency for those who have been late with repayments in the past, whereas those who never fall behind on payments are extremely unlikely to become delinquent, as shown by the almost flat boxplot (fig.5).

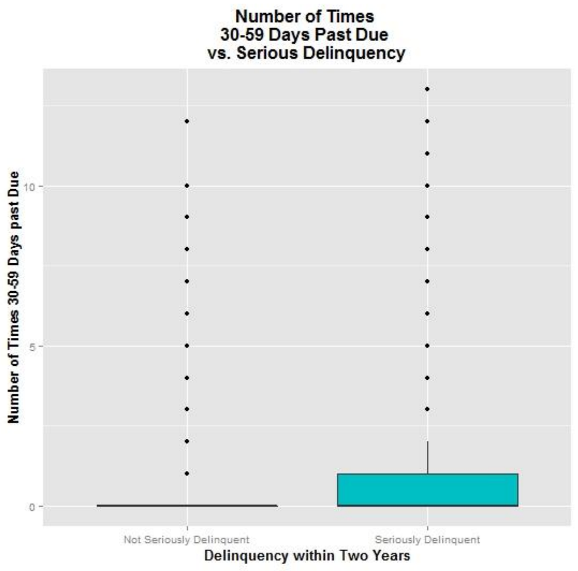

Figure 5

The next step is to split the original training data in such a way that (say) 70% is randomly allocated to the train set on which the model will be built and the rest held back to form the validation set (which will subsequently be used to evaluate the models). Note that the split was executed in such a way that the same original proportions of serious delinquents and non-delinquents are allocated to each set, since otherwise bias might be introduced simply because the individual train and validation sets are not adequately representative of the combined data.

Decision Tree Model

Having cleaned the dataset, modelling can begin. There are many techniques for solving these dichotomous problems, and here a decision tree was tried first. Classification and regression tree (CART) models are popular because they are easy to interpret — predictions can be made for each new case simply by following a series of clear if-then rules. In the case of a classification tree, the tree algorithm works by iteratively splitting the dataset according to how homogenous the data is relative to the target variable [5].

In the tree produced using the rpart package *(fig. 6)*, the most homogenous component was that 93% of loans in which at least one 90 or more days late were non-delinquent, and so forth until the model's stopping rules are met.

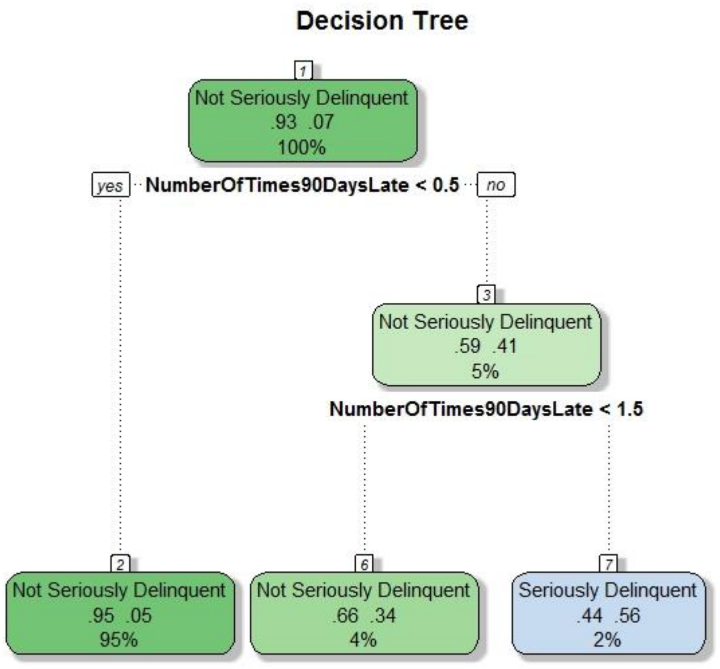

Figure 6

Starting at the top of the tree, the first root node (labelled 1), shows that 7% of loans ended in serious delinquency. The first split is then made by the decision tree, based on the number of times 90 days late variable. If this is greater than or equal to 0.5 (i.e. one or more, since this is an integer) then the serious delinquency rate is 41% (node 3), whereas otherwise it is just

5% (node 2). The model makes no further splits on the left-hand side (node 2 is a "terminal" node).

Returning to the right-hand side of the tree, only one further split is made. Among those who have been 90 days late at least once, the ones who have been late more than once are more at risk, with a 56% serious delinquency rate compared to only 34% for those who have been that late only once.

This model suggests that just knowing whether or not someone has been 90 days late with a repayment either never, once, or more than once is a good indicator of whether they may become seriously delinquent. But how well does it make predictions? After running it against the validation data a classification (or "confusion" matrix) can be produced from the results *(fig. 7)*. Model accuracy can be calculated by summing the correct predictions and dividing by the total number of observations. This gives 93.49% (42,069/45,000), slightly better than the baseline rate of 93.32% (which can be regarded here simply as the accuracy rate if you had simply guessed "non-delinquent" for every prediction).

CONFUSION MATRIX FOR DECISION TREE			
		Predicted	
		Non-Delinquent	Delinquent
Actual	Non-Delinquent	41597	395
	Delinquent	2536	472

Figure 7

Clearly there are some issues with this model. It lacks "precision", hitting on just 54.4% (472/ 867) predictions of serious delinquency, with a true positive rate (or "sensitivity") of just 472/3008 (15.7%). The model is poor at identifying actual serious delinquents, failing to provide the lender with accurate information on where their risk lies. In addition, the classification model yes/no approach doesn't allow any implementation flexibility — no probabilities are produced so (unlike in logistic regression, for example) there is no opportunity to adjust the "threshold" to allow for the fact that one kind of prediction error may be more costly than another.

Logistic Regression Model

Another common approach to binary classification is the logistic regression ("or logit") model, which predicts the *probability* of an event occurring depending on values of the explanatory variables. The regression coefficients for the predictors are calculated using maximum likelihood estimation, but unlike CART, logistic regression assumes a linear model, which may well not be appropriate here.

The first attempt at a logit model, trained on all of the explanatory variables (trainLog), runs into a complete separation issue [6]. The model does not converge because there are 18 clients with monthly income greater than 250,000, none of whom are seriously delinquent. The method chosen to resolve this was to split income into bins (five were chosen: below 1000, 1000-4999, 5000-9999, 10000-49999 and 50000 or more), treating it as a categorical rather than continuous variable.

In the revised model (trainLog1) every explanatory variable is significant at the 95% level except for revolving utilisation of unsecured lines and number of open credit lines and loans. It is worth considering whether one or more of those variables should be removed from the model, to reduce the chance of "overfitting" the training data, whereby the model fails to distinguish between the true "signal" of the underlying relationships and random noise.

One approach to model reduction is to remove the variable with the least significance and repeat the process until only "significant" variables are left. Another is to automate the process using stepwise regression based on some criterion. The Akaike Information Criterion (AIC), which builds a parsimonious model by penalising models that utilise more variables [7], was used here.

The AIC of the "full" model (trainLog1) is 41970. Can this be reduced with, say, a backwards stepwise regression model (trainLog2)? By removing the revolving utilisation of unsecured lines and number of open credit lines and loans from the model, AIC can indeed be reduced to 41967. However, an ANOVA test on the two models produced a (barely) significant chi-square value (0.4658), suggesting that the second model does not fit as well (although for simplicity's sake it will be retained here).

It is also possible to perform forwards stepwise regression, starting with just the intercept, and this process may actually produce a different "best" model. Furthermore, stepwise regression does not consider every possible model, so the "optimal" model may even be overlooked [8].

Whereas the CART model produced an easy-to-understand decision tree, the coefficients from a logistic regression model are harder to interpret. To make a new prediction it is first necessary to input the values into the formula before converting the output into a probability. For example, let us consider a loan relating to a 40-year-old with one dependent and a monthly income of 4000 (with every other variable set to zero).

$$\text{prediction} = -1.808513 + (40\times-0.02768499) + (1\times 0.04225236) + (0.03647074)$$
$$\text{prediction} = -2.83719$$
$$\text{odds} = \exp(\text{prediction}) = 0.0585901$$
$$\text{probability} = \frac{odds}{odds+1} = 0.0553473$$

Thus there is roughly a 5.5% chance that the loan will become seriously delinquent.

By taking the exponent of the coefficients, it is possible to see how the odds of serious delinquency are affected by a change in one of the predictor variables (assuming that all others are held constant). Thus those odds are reduced by a factor of 0.973 for every one year increase in age, but increase by a factor of 2.361 for every time that someone has been more than 90 days late etc.

Compared to the decision tree, this model (assuming a "standard" 0.5 threshold) produces a slight improvement in accuracy, much better precision, and slightly worse sensitivity. However, that 0.5 threshold assumes that false positives ("Type 1" errors) and false negatives ("Type 2" errors) are equally costly. In the case of loans, this is not likely to be the case — false negatives (occasions when the model failed to predict serious delinquency that *did* occur) may be much more costly than false positives (when it predicted serious delinquency that *did not* occur, i.e. false alarms).

Given that (like the decision tree) this model underrepresents incidence of serious delinquency, it may be circumspect to lower the threshold so that more actual delinquency incidents are identified (higher sensitivity), even at the cost of worse specificity due to more false positives. Reducing the threshold to 0.2 here doubles the chance of correctly identifying actual delinquency with only a small trade-off in lost accuracy *(fig.8).* Ideally, however, the optimal threshold would be determined by someone with knowledge of the relative costs of these errors.

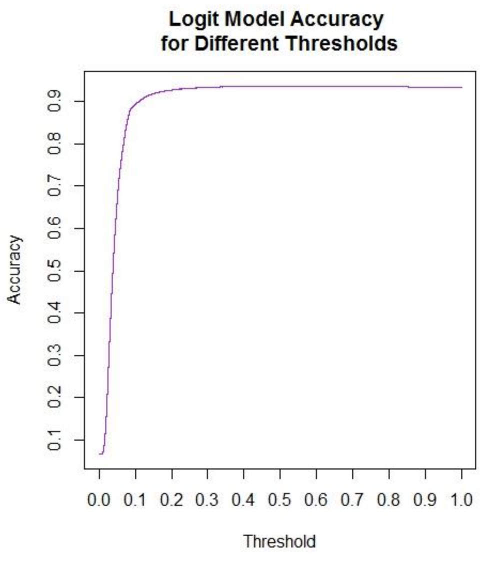

Figure 8

Instead of accuracy, which is not a robust metric for unbalanced data, the Kaggle competition used an absolute measure of the quality of predictions, the "area under the curve" (AUC). The AUC is derived from a Receiver Operating Characteristic (ROC) curve, which typically organises classifiers by plotting the true positive rate against the false positive rate. The ROC curve captures all thresholds simultaneously (and can thus assist with threshold selection) [9]. An AUC of 1 would be representative of perfect classification between high- and low-risk loans, whereas an AUC below 0.5 indicates that simply guessing would have been more effective. The competition winner achieved an AUC of 0.86956, but the decision tree and logit models here only managed 0.64829 and 0.81289 respectively on the exact same data.

Additional Models and Summary

In view of this poor performance, two alternative models were built: a gradient boosted machine (GBM) and a random forest. Each of these works by building ensembles of decision trees. Although more powerful than simple decision trees, they are much harder to interpret. Interestingly, however, the variable importance output of both new models suggests that revolving utilisation of unsecured lines (which was discarded using stepwise logit) is one of the key predictors. These models may perhaps be capturing some interactions that eluded the logit model.

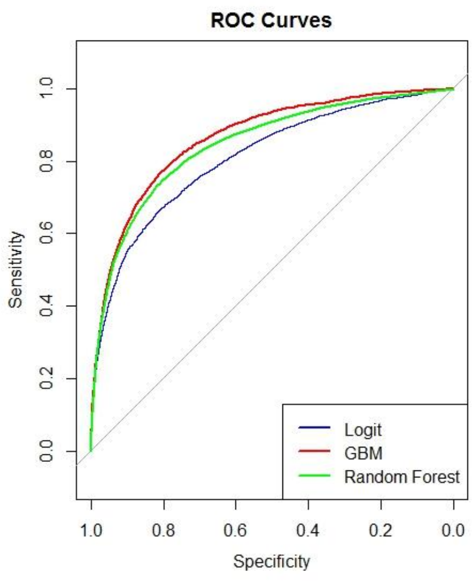

Figure 9

Looking at the ROC curves *(fig.9)*, the new models both perform well across a range of thresholds. The (blue) logit curve lies beneath the two other curves, indicating lower AUC. (N.B. The pROC package in R plots *specificity* on the x-axis rather than the false positive rate, i.e. *1 – specificity*, which means that the x-axis runs down from 1.0 in rather unintuitive fashion.

Here is a summary of the performance of all four models *(fig.10)*:

	METRIC					
TYPE OF MODEL	**Accuracy**	**Sensitivity**	**Specificity**	**AUC**	**Kaggle AUC***	**Kaggle Position***
Classification Tree	0.93487	0.15691	**0.99059**	N/A	0.64829	868th
Logit (0.2 threshold)	0.92753	0.31848	0.97116	0.80964	0.81289	756th
GBM	***0.93651***	***0.57197***	0.94521	**0.863**	**0.86413**	**417th**
Random Forest	0.93558	0.56720	0.94234	0.8462	0.85335	635th

Figure 10 (Model Comparison)

** Kaggle results relate to the private leaderboard (for which submissions can still be assessed even though the competition has closed). This leaderboard contained 925 entries [10].*

References

[1] Kaggle, “Give Me Some Credit”, available from <https://www.kaggle.com/c/GiveMeSomeCredit>.
[2] “Credit score”, (wiki article). Available from <https://en.wikipedia.org/wiki/Credit_score>.
[3] Kaggle data available from <https://www.kaggle.com/c/GiveMeSomeCredit/data>.
[4] Kabacoff, R. (2015) *R in Action (2nd edition)*, Manning, p 428.
[5] Linoff G.S. & Berry, M.J.A. (2011) *Data Mining Techniques (3rd edition)*, Wiley p 237.
[6] Albert A. & Anderson J.A. (1984) “On the existence of maximum likelihood estimates in logistic regression models”, *Biometrika* 71: pp 1-10.
[7] *Ibid.* 4, p 202.
[8] *Ibid.* p 204.
[9] *Ibid.* p 408.
[10] Kaggle leaderboard from <https://www.kaggle.com/c/GiveMeSomeCredit/leaderboard>.

Appendix 1 — Reflections

This project presented some interesting challenges. Initial analysis revealed some missing values and coding issues, and also that there was a severe imbalance within the target variable, which could result in model bias towards the majority class.

In order to predict which loans were higher risk, binary classification modelling was used. Credit scores were generated for each loan, based on the probability of it becoming seriously delinquent within two years.

After preparing the data, various different models were tried with mixed results. The classification decision tree was easy to interpret, but was poor at identifying actual incidence of serious delinquency and failed to generalise well when applied to new data. Although the logit model achieved better AUC performance, it lacked the intuitive interpretability of the decision tree. (Both of these models also underrepresented the actual quantity of serious delinquency incidence in their predictions, indicating class bias.) The GBM and random forest models were even more difficult to interpret, since they are essentially “black box” techniques, but each produced far superior performance, which could possibly even be enhanced by some parameter tuning.

Access to domain expertise would also be extremely useful. A credit scoring expert would be able to fully dissect the structure found in each variable, identify which outliers might be erroneous data, and advise ways in which predictors might be combined to generate useful derived variables. In order to assess the potential value of this, a new variable was generated by dividing income by family size (the logic being that larger families might possibly have relatively less discretionary income), which resulted in slightly better GBM performance, an AUC of 0.86445 on the Kaggle competition test set.

More sophisticated models would incorporate some mechanism for offsetting target variable imbalance, perhaps by oversampling the minority class (or possibly undersampling the majority class, although that would involve throwing out data) to balance the dataset. This oversampling would typically involve bootstrapping the minority class to generate many more observations sourced from the existing ones (bootstrapping is a form of random sampling with replacement). Another technique for dealing with unbalanced data is to adjust the costs of misclassification, so that the penalty for missing actual incidence of delinquent loans is higher, but changing the threshold becomes academic if the goal is simply to maximise AUC (as was the case in this competition).

Another common technique for potentially enhancing performance is to create an ensemble from multiple models, combining their predictions using an averaging process. Models often produce better predictions when combined than they do individually (indeed, data competitions are frequently won by people or teams using ensembles). A simple average of the predictions from the GBM and random forest models from this project resulted in an improvement in AUC to 0.86534, good for 221st place and within .005 of the winner’s score. Naturally, in the financial world improving AUC by just a fraction might be worth tens of thousands of pounds.

With regards other applications of these models, none of the other project proposals from the course clearly lend themselves to binary classification analysis. However, these modelling techniques are common when the target variable is dichotomous, in areas of medicine (e.g. predicting whether a tumour is malignant or benign), customer relationship management (e.g. predicting whether or not someone will renew a mobile phone contract), polling (e.g. predicting whether someone will vote Republican or Democrat), parole decisions (predicting whether or not someone will violate their parole) etc.

Appendix 2 — R Code

```
# Packages required: nortest, mice, ggplot2, caTools, rpart, rpart.plot, rattle, MASS, ROCR, caret,
# class, e1071, pROC, gbm.

# First download data from https://www.kaggle.com/c/GiveMeSomeCredit/data and set the
# working directory to the folder where those data are stored. In my case this is
setwd("C:/Program Files/R/R_Working_Directory/Kaggle_Credit")
# but that directory will of course be different for others.

# Now read in both the training and test sets supplied by Kaggle:
credit <- read.csv("cs-training.csv")
cs_test <- read.csv("cs-test.csv")

### DATA UNDERSTANDING ###

# Initial look at training set:
str(credit)

# Here we see that there are 150000 observations (rows) with 12 variables (columns). The first
# column, "X", is just a row number, and we then have the dependent (or "target") variable
# "SeriousDlqin2yrs" followed by the ten dependent (or # "explanatory") variables.

### EXPLORATORY DATA ANALYSIS (Summaries and Visualisations)
summary(credit)

# Analyse target variable (Serious Delinquency within two years)
mean(credit$SeriousDlqin2yrs) # mean is only 0.06684, so only 6.7% of loans resulted in
# delinquency.
table(credit$SeriousDlqin2yrs) # 139974 non-delinquents vs. 10026 delinquents so this is an
# "unbalanced" dataset.
# Bar chart of target variable (FIG. 1)
ylim <- c(0, 1.1*max(table(credit$SeriousDlqin2yrs))) # create space for text
bp <- barplot(table(credit$SeriousDlqin2yrs), main="Bar Chart of Seriously Delinquent Loans",
            xlab="Loan Outcome", ylab="Number of Loans", ylim=ylim,
            names.arg=c("Non-Delinquent", "Delinquent"), col=c("blue", "red"))
text(x=bp, y=table(credit$SeriousDlqin2yrs), label=table(credit$SeriousDlqin2yrs), cex=1, pos=3)

# Age
table(credit$age)

# Side-by-side plots for age variable (FIG. 2)
par(mfrow=c(1,2)) # set up side-by-side plot display
hist(credit$age, main="Histogram of \nBorrower's Age",
    xlab="Age", col="brown")
qqnorm(credit$age, main="Normal Q-Q Plot \nof Borrower's Age", col="brown")
par(mfrow=c(1,1)) # return to single plot display

# Test age for normality
library(nortest)
lillie.test(credit$age)

# Monthly income
table(credit$MonthlyIncome > 1000000/12)

# Side-by-side plot of monthly income (FIG. 3)
par(mfrow=c(1,2)) # set up side-by-side plot display
hist(credit$MonthlyIncome, main="Histogram of \nMonthly Income",
    xlab="Income", col="gold3") # FIG. 3
qqnorm(credit$MonthlyIncome, main="Normal Q-Q Plot \nof Monthly Income", col="gold3")
```

```
# Boxplot of monthly income (FIG .4)
boxplot(credit$MonthlyIncome, main="Standard R Boxplot \nof Monthly Income \n(with outliers)",
       ylab="Monthly Income", col="purple")
boxplot(credit$MonthlyIncome, main="Boxplot of Monthly \nIncome (outliers removed)",
       ylab="Monthly Income", outline=F, col="green")
par(mfrow=c(1,1)) # return to single plot display

# Number of open credit loans or lines
hist(credit$NumberOfOpenCreditLinesAndLoans)  # heavily right-skewed
table(credit$NumberOfOpenCreditLinesAndLoans)

# Number of dependents
hist(credit$NumberOfDependents)
table(credit$NumberOfDependents)

# Number of real estate loans or lines
hist(credit$NumberRealEstateLoansOrLines)
table(credit$NumberRealEstateLoansOrLines)

# Revolving utilization of unsecured lines
hist(credit$RevolvingUtilizationOfUnsecuredLines) # heavily right-skewed
table(credit$RevolvingUtilizationOfUnsecuredLines > 10)

# Debt ratio
hist(credit$DebtRatio) # heavily right-skewed
table(credit$DebtRatio > 10000)

# 30-59, 60-89 and 90+ days late
table(credit$NumberOfTime30.59DaysPastDueNotWorse)
table(credit$NumberOfTime60.89DaysPastDueNotWorse)
table(credit$NumberOfTimes90DaysLate) # All three tables show the same outliers: exactly 5 values
```

```
# of 96 and 264 values of 98.
hist(credit$NumberOfTime30.59DaysPastDueNotWorse) # Right-skewed

### DATA PREPARATION ###

# Merge training and testing datasets for combined preparation
credit_merged <- rbind(credit, cs_test)

# Review combined dataset
str(credit_merged)
summary(credit_merged)

# Replace 96 and 98 values in 30-59, 60-89 and 90+ days late columns by replacing them with
# median
credit_merged$NumberOfTime30.59DaysPastDueNotWorse[credit_merged$
    NumberOfTime30.59DaysPastDueNotWorse == 96 |
      credit_merged$NumberOfTime30.59DaysPastDueNotWorse == 98] <-
        median(credit_merged$NumberOfTime30.59DaysPastDueNotWorse)
credit_merged$NumberOfTime60.89DaysPastDueNotWorse[credit_merged$
    NumberOfTime60.89DaysPastDueNotWorse == 96 |
      credit_merged$NumberOfTime60.89DaysPastDueNotWorse == 98] <-
        median(credit_merged$NumberOfTime60.89DaysPastDueNotWorse)
credit_merged$NumberOfTimes90DaysLate[credit_merged$NumberOfTimes90DaysLate == 96 |
      credit_merged$NumberOfTimes90DaysLate == 98] <-
        median(credit_merged$NumberOfTimes90DaysLate)

# Deal with erroneous "zero" age values (replace with median age value)
median(credit_merged$age)
credit_merged$age[credit_merged$age==0] <- median(credit_merged$age)
min(credit_merged$age) # here we see that the minimum age is now 21.
```

```
# Deal with many thousand missing values for monthly income and number of dependents using
# multiple imputation
library(mice)
md.pattern(credit_merged) # missing 6550 no of dependents and 49834 monthly income values
# across merged dataset.
vars_for_imputation <- credit_merged[c("MonthlyIncome", "NumberOfDependents")] # The two
# variables that require imputation.
set.seed(180) # This sets the seed for the random number generator so that results will be
# reproducible.
# ***WARNING - The next step takes a very long time to process; once it has been completed it is
# easiest just to load the new values via a csv file.***
imputed <- complete(mice(vars_for_imputation))
write.csv(imputed, "imputed_export.csv") # saves imputed values as csv file for later use.
# imputed <- read.csv("imputed_export.csv", header=T) # run this if/when importing the new
# csv file (which can be supplied if required).
credit_merged$MonthlyIncome <- imputed$MonthlyIncome # incorporates imputed income values
# to dataframe.
credit_merged$NumberOfDependents <- imputed$NumberOfDependents # incorporates imputed
# no of dependents values to dataframe.

# Review dataset
summary(credit_merged) # Here we see that some of the columns have been tidied up, with all
# missing values removed except for the 101,289 target variable values that we are trying
# to predict.

# split dataset back into original two components, training set and test set.
credit <- credit_merged[1:150000,]
cs_test <- credit_merged[150001:251503,]

# Construct correlation matrix.
ctab <- cor(credit)
ctab <- round(ctab, 2) # Round to two decimal places for easier viewing.
ctab


# Look at sd of each variable
apply(credit, 2, sd)


# For use with plot below
median(credit$NumberOfTime30.59DaysPastDueNotWorse[credit$SeriousDlqin2yrs==1])
median(credit$NumberOfTime30.59DaysPastDueNotWorse[credit$SeriousDlqin2yrs==0])
mean(credit$NumberOfTime30.59DaysPastDueNotWorse[credit$SeriousDlqin2yrs==1])
mean(credit$NumberOfTime30.59DaysPastDueNotWorse[credit$SeriousDlqin2yrs==0])


# Turn dependent variable into factor for modelling.
credit$SeriousDlqin2yrs <- factor(credit$SeriousDlqin2yrs, labels=c("Not Seriously Delinquent",
                        "Seriously Delinquent"))


# More advanced boxplot (FIG. 5)
library(ggplot2)
ggplot(credit, aes(x = SeriousDlqin2yrs, y = NumberOfTime30.59DaysPastDueNotWorse,
                fill = SeriousDlqin2yrs)) + geom_boxplot() +
      labs(title="Number of Times \n30-59 Days Past Due \nvs. Serious Delinquency",
          x="Delinquency within Two Years",
          y="Number of Times 30-59 Days past Due", face="bold") +
            theme(plot.title = element_text(size=16, face="bold", vjust=1)) +
              # modify title header
                  theme(axis.title = element_text(size=13, face="bold")) +
              # modify x-axis header
                        guides(fill=FALSE) # remove legend


# Split data into training set and validation set for modelling using caTools package
library(caTools)
```

```
set.seed(180)
split <- sample.split(credit$SeriousDlqin2yrs, SplitRatio = 0.7) # split using target variable as
# base
train <- subset(credit, split == TRUE)
val <- subset(credit, split == FALSE)

### DECISION TREE MODEL ###

# Create model using rpart package
library(rpart)
trainTree <- rpart(SeriousDlqin2yrs ~ .-X, data=train)
summary(trainTree)

# Plot decision tree (FIG.6)
library(rpart.plot)
library(rattle)
fancyRpartPlot(trainTree, main="Decision Tree")

# Establish baseline accuracy rate
table(val$SeriousDlqin2yrs)
41992/nrow(val) # 93.32% of these loans were non-delinquent (baseline).

# Make predictions on (unseen) validation data
val$predicted_risk_Tree <- predict(trainTree, newdata=val, type="class")
summary(val$predicted_risk_Tree)
table(val$predicted_risk_Tree)
867/nrow(val) # only 2% delinquency rate so model is underrepresenting delinquency rate
treeTable <- table(val$SeriousDlqin2yrs, val$predicted_risk_Tree, dnn=c("Actual", "Predicted"))
treeTable

# Assess performance using function from R in Action (Kabacoff, p.406)
```

```
model_performance <- function(table, n=5){
    if(!all(dim(table) == c(2,2)))
        stop("Must be a 2 x 2 table")
    tn = table[1,1]
    fp = table[1,2]
    fn = table[2,1]
    tp = table[2,2]
    sensitivity = tp/(tp+fn)
    specificity = tn/(tn+fp)
    ppp = tp/(tp+fp)
    npp = tn/(tn+fn)
    hitrate = (tp+tn)/(tp+tn+fp+fn)
    result <- paste("Sensitivity = ", round(sensitivity, n) ,
            "\nSpecificity = ", round(specificity, n),
            "\nPositive Predictive Value = ", round(ppp, n),
            "\nNegative Predictive Value = ", round(npp, n),
            "\nAccuracy = ", round(hitrate, n), "\n", sep="")
    cat(result)
}

model_performance(treeTable)

### LOGISTIC REGRESSION MODEL ###

# First attempt - complete separation issue
trainLog <- glm(SeriousDlqin2yrs ~ . -X, data=train, family=binomial)
# "Warning message: glm.fit: fitted probabilities numerically 0 or 1 occurred"

# Identify and resolve issue
table(train$SeriousDlqin2yrs, train$MonthlyIncome > 250000)
train$MonthlyIncome_cat <- cut(train$MonthlyIncome, breaks = c(0, 1000, 5000, 10000,
                                    50000, 10000000), include.lowest=TRUE)
val$MonthlyIncome_cat <- cut(val$MonthlyIncome, breaks =  c(0, 1000, 5000, 10000,
                                    50000, 10000000), include.lowest=TRUE)
table(train$MonthlyIncome_cat)

# New logistic regression model with income in bins
trainLog1 <- glm(SeriousDlqin2yrs ~ .-X -MonthlyIncome,
              data=train, family=binomial)
summary(trainLog1)

# Stepwise regression
library(MASS)
trainLog2 <- stepAIC(trainLog1, direction="backward")
trainLog2

# View coefficients
coef(trainLog2)
# (Intercept)                          age NumberOfTime30.59DaysPastDueNotWorse
# -1.808513e+00                   -2.768499e-02                    5.703678e-01
# DebtRatio             NumberOfTimes90DaysLate        NumberRealEstateLoansOrLines
# -4.186091e-05                   8.592803e-01                    1.041325e-01
# NumberOfTime60.89DaysPastDueNotWorse                NumberOfDependents
MonthlyIncome_cat(1e+03,5e+03]
# 7.574588e-01                   4.225236e-02                    3.647074e-02
# MonthlyIncome_cat(5e+03,1e+04]       MonthlyIncome_cat(1e+04,5e+04]
MonthlyIncome_cat(5e+04,1e+07]
# -2.466548e-01                   -4.768740e-01                   -3.284278e-01

# Create example prediction
prediction <- -1.808513+(40*-0.02768499)+(1*0.04225236)+(0.03647074)
prediction # -2.83719
```

```
# Convert to probability
odds <- exp(prediction)
odds # 0.0585901
probability <- odds/(odds + 1)
probability # 0.0553473

# Odds ratio
exp(coef(trainLog2))
exp(confint(trainLog2)) # 95% confidence intervals for the coefficients (on an odds scale).

# compare to trainLog1 using ANOVA
anova(trainLog2, trainLog1, test="Chisq") # 0.4658 so statistically significant difference in fit.

# Make predictions on (unseen) validation data
val$predicted_risk_Log2 <- predict(trainLog2, type="response", newdata=val)
summary(val$predicted_risk_Log2)
table(val$predicted_risk_Log2 > 0.5)
777/nrow(val) # again less than 2% of loans are being predicted as serious delinquency
log2Table_0.5 <- table(val$SeriousDlqin2yrs, val$predicted_risk_Log2 > 0.5, dnn=c("Actual",
                                                  "Predicted"))
log2Table_0.5
model_performance(log2Table_0.5)

# Plot accuracy vs. threshold (# FIG.8)
plot(performance(ROCRLog2, measure = "acc"), col="purple", xlab="Threshold", main=
    "Logit Model Accuracy \nfor Different Thresholds")
axis(1, at=c(0.1, 0.3, 0.5, 0.7, 0.9))
axis(2, at=c(0.1, 0.3, 0.5, 0.7, 0.9))

# Change threshold to 0.2
log2Table_0.2 <- table(val$SeriousDlqin2yrs, val$predicted_risk_Log2 > 0.2, dnn=c("Actual",
                                   "Predicted"))
log2Table_0.2
model_performance(log2Table_0.2)

# Percentage of predictions that were predicted serious delinquency
(958+1211)/nrow(val) # 0.0482

# AUC
library(ROCR)
ROCRLog2 <- prediction(val$predicted_risk_Log2, val$SeriousDlqin2yrs)
aucLog2 <- as.numeric(performance(ROCRLog2, "auc")@y.values)
aucLog2 # 0.8096439

# Prepare ROC performance for subsequent gbm plot
perf_Log2 <- performance(ROCRLog2, "tpr", "tnr")

### GBM MODEL ### *** WARNING — THIS MAY TAKE SEVERAL HOURS TO PROCESS

# load caret and other required packages
library(caret)
library(class)
library(e1071)
library(pROC)
library(gbm)

train$MonthlyIncome_cat <- NULL
val$MonthlyIncome_cat <- NULL
levels(train$SeriousDlqin2yrs) <- c("No", "Yes")
levels(val$SeriousDlqin2yrs) <- c("No", "Yes")
gbmControl <- trainControl(method='cv', number=3, returnResamp='none',
             summaryFunction = twoClassSummary, classProbs = TRUE)
```

```
set.seed(180)

gbm <- train(SeriousDlqin2yrs ~ .-X, data=train,

        method="gbm",

        trControl=gbmControl,

        metric = "ROC",

        preProc = c("center", "scale"), verbose = FALSE)

summary(gbm) # which variables were most important to the model?

plot(varImp(gbm,scale=F)) # plot variable importance

ggplot(gbm) # plots ROC vs. boosting iterations of the three training models

print(gbm) # find out Which tuning parameters were most important to the model


# Evaluate

val$gbm_predictions <- predict(gbm, val, type="raw")

postResample(val$gbm_predictions, val$SeriousDlqin2yrs) # 0.9365111 accuracy

gbmTable <- table(val$gbm_predictions, val$SeriousDlqin2yrs, dnn=c("Actual", "Predicted"))

gbmTable

model_performance(gbmTable) # poor precision (positive predictive value) but model is making

# many more predictions of serious deliquency occurring than tree and logit, as shown here:


# Percentage of predictions that were predicted serious delinquency

(600+2408)/nrow(val) # 0.06684444


# Now look at probabilities:

gbm_predictions <- predict(object=gbm, val, type="prob")

val$gbm_predictions_prob <- gbm_predictions[,2]

gbm_ROC <- roc(predictor=val$gbm_predictions_prob,

         response=val$SeriousDlqin2yrs,

         levels=c("No", "Yes"))

gbm_ROC #  AUC is 0.863


### RANDOM FOREST ### *** WARNING — THIS MAY TAKE SEVERAL HOURS TO PROCESS
```

```
rfControl <- trainControl(method = "cv", number = 3, returnResamp="none",
                    summaryFunction = twoClassSummary, classProbs = TRUE)
set.seed(180)
rf <- train(SeriousDlqin2yrs ~ .-X, data=train,
          method="rf",
          trControl=rfControl,
          metric = "ROC",
          preProc = c("center", "scale"))
plot(varImp(rf,scale=F)) # plot variable importance
print(rf) # find out Which tuning parameters were most important to the model (tuning may
# improve performance)

# Evaluate
val$rf_predictions <- predict(rf, val, type="raw")
postResample(val$rf_predictions, val$SeriousDlqin2yrs) # accuracy of 0.9355778
rfTable <- table(val$rf_predictions, val$SeriousDlqin2yrs, dnn=c("Actual", "Predicted"))
rfTable
model_performance(rfTable)

# Percentage of predictions that were predicted serious delinquency
(460+2548)/nrow(val) # 0.06684444

# Now look at probabilities:
rf_predictions <- predict(rf, val, type="prob")
val$rf_predictions_prob <- rf_predictions[,2]
rf_ROC <- roc(predictor=val$rf_predictions_prob,
           response=val$SeriousDlqin2yrs,
           levels=c("No", "Yes"))
rf_ROC #  AUC is 0.8462

# Plot ROC curves
```

```
plot(gbm_ROC, col="red", main="ROC Curves" )
plot(perf_Log2, add=TRUE, col = 'blue')
plot(rf_ROC, add=TRUE, col = 'green')
legend("bottomright", legend=c("Logit", "GBM", "Random Forest"),
    col=c("blue", "red", "green"), lwd=2) # FIG.9

### GBM with additional monthly income/family size ### *** WARNING - THIS MAY TAKE AGES
family_size_train <- train$NumberOfDependents + 1 # create family size variable
family_size_val <- val$NumberOfDependents + 1
train$income_family_size <- train$MonthlyIncome/family_size_train
val$income_family_size <- val$MonthlyIncome/family_size_val
set.seed(180)
gbm2 <- train(SeriousDlqin2yrs ~ .-X, data=train,
        method="gbm",
        trControl=gbmControl,
        metric = "ROC",
        preProc = c("center", "scale"), verbose = FALSE)
summary(gbm2) # which variables were most important to the model?
ggplot(gbm2) # plots ROC vs. boosting iterations of the three training models
plot(varImp(gbm2,scale=F)) # plot variable importance
print(gbm2) # find out Which tuning parameters were most important to the model

# Evaluate
val$gbm2_predictions <- predict(gbm2, val, type="raw")
postResample(val$gbm2_predictions, val$SeriousDlqin2yrs) # 0.9365111 accuracy
gbm2Table <- table(val$gbm2_predictions, val$SeriousDlqin2yrs, dnn=c("Actual", "Predicted"))
gbm2Table
model_performance(gbm2Table) # poor precision (positive predictive value) but model is making
# many more predictions of serious deliquency occurring than tree and logit, as shown here:

# Percentage of predictions that were predicted serious delinquency
```

```
(607+2401)/nrow(val) # 0.06684444


# Now look at probabilities:
gbm2_predictions <- predict(object=gbm2, val, type="prob")
val$gbm2_predictions_prob <- gbm2_predictions[,2]
gbm2_ROC <- roc(predictor=val$gbm2_predictions_prob,
            response=val$SeriousDlqin2yrs,
            levels=c("No", "Yes"))
gbm2_ROC #  AUC is 0.863



### SUBMISSIONS TO KAGGLE ###
# Edit test dataframe for consistency with training data
cs_test$SeriousDlqin2yrs <- factor(cs_test$SeriousDlqin2yrs, levels = c(0,1))


# create decision tree entry
cs_test$predicted_risk_Tree <- predict(trainTree, type ="prob", cs_test)
submissionTree <- data.frame(Id = seq(1:101503), Probability = cs_test$predicted_risk_Tree)
names(submissionTree) <- c("Id", "Unused", "Probability")
write.csv(submissionTree[c(1,3)], "submissionTree.csv", row.names=FALSE) # 0.648290 (868th)


# create stepwise logistic regression entry
cs_test$MonthlyIncome_cat <- cut(cs_test$MonthlyIncome, breaks =  c(0, 1000, 5000, 10000,
                                          50000, 10000000), include.lowest=TRUE)
cs_test$predicted_risk_Log2 <- predict(trainLog2, type ="response", cs_test)
submissionLog2 <- data.frame(Id = seq(1:101503), Probability = cs_test$predicted_risk_Log2)
write.csv(submissionLog2, "submissionLog.csv", row.names=FALSE) # 0.812894 (756th)
cs_test$MonthlyIncome_cat <- NULL


# create GBM entry
cs_test$predicted_risk_GBM <- predict(gbm, type ="prob", cs_test)
```

```
submissionGBM <- data.frame(Id = seq(1:101503), Probability = cs_test$predicted_risk_GBM)
names(submissionGBM) <- c("Id", "Unused", "Probability")
write.csv(submissionGBM[c(1,3)], "submissionGBM.csv", row.names=FALSE) # 0.86413 (417th place)

# create RF entry
cs_test$predicted_risk_RF <- predict(rf, type ="prob", cs_test)
submissionRF <- data.frame(Id = seq(1:101503), Probability = cs_test$predicted_risk_RF)
names(submissionRF) <- c("Id", "Unused", "Probability")
write.csv(submissionRF[c(1,3)], "submissionRF.csv", row.names=FALSE) # 0.853354 (635th place)

# create GBM entry with additional derived variable
family_size_cs_test <- cs_test$NumberOfDependents + 1 # create family size variable
cs_test$income_family_size <- cs_test$MonthlyIncome/family_size_cs_test
cs_test$predicted_risk_GBM2 <- predict(gbm2, type ="prob", cs_test)
submissionGBM2 <- data.frame(Id = seq(1:101503), Probability = cs_test$predicted_risk_GBM2)
names(submissionGBM2) <- c("Id", "Unused", "Probability")
write.csv(submissionGBM2[c(1,3)], "submissionGBM2.csv", row.names=FALSE) # 0.864452
# (321st place)

# create ensemble entry
submission_Ens <- cbind(submissionGBM[c(1,3)], submissionRF[3])
submission_Ens$merge <- (submission_Ens[,2] + submission_Ens[,3])/2
names(submission_Ens) <- c("Id", "Unused1", "Unused2", "Probability")
write.csv(submission_Ens[c(1,4)], "submissionEns.csv", row.names=FALSE) # 0.865339 (221st place)
```