FINAL PROJECT

Data Analytics and Modelling

Kothari, Aman

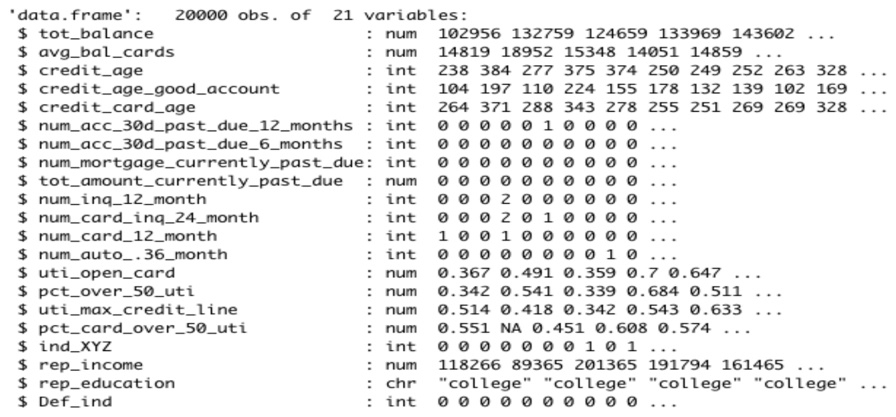
Virginia Tech

Objective

The purpose of this project is to demonstrate the visualization and analytical techniques learned in Data Analytics and Visualization class offered at Virginia Tech. I will use the dataset provided to us containing historical data on response(binary) and 20 predictor variables from credit card accounts for a hypothetical bank XYZ. The techniques involved are data visualization, logistical regression, and neural network modelling and the project is divided into the same categories.

Data Visualization (Part 1)

The dataset has been provided to us in two parts. One is the training dataset with 20000 observations and the other is the testing dataset with 5000 observations. The response variable is identified by ***Def_Ind*** while the rest of the variables are predictors. Once the training dataset is loaded into R, we can use the str() function to view the initial data. This initial data is not indicative of the entire dataset, but it gives a good approximation of what the data may look like.



```
'data.frame':   20000 obs. of  21 variables:
 $ tot_balance                   : num  102956 132759 124659 133969 143602 ...
 $ avg_bal_cards                 : num  14819 18952 15348 14051 14859 ...
 $ credit_age                    : int  238 384 277 375 374 250 249 252 263 328 ...
 $ credit_age_good_account       : int  104 197 110 224 155 178 132 139 102 169 ...
 $ credit_card_age               : int  264 371 288 343 278 255 251 269 269 328 ...
 $ num_acc_30d_past_due_12_months : int  0 0 0 0 0 1 0 0 0 0 ...
 $ num_acc_30d_past_due_6_months  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ num_mortgage_currently_past_due: int  0 0 0 0 0 0 0 0 0 0 ...
 $ tot_amount_currently_past_due  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ num_inq_12_month              : int  0 0 0 2 0 0 0 0 0 0 ...
 $ num_card_inq_24_month         : int  0 0 0 2 0 1 0 0 0 0 ...
 $ num_card_12_month             : int  1 0 0 1 0 0 0 0 0 0 ...
 $ num_auto_.36_month            : int  0 0 0 0 0 0 0 0 1 0 ...
 $ uti_open_card                 : num  0.367 0.491 0.359 0.7 0.647 ...
 $ pct_over_50_uti               : num  0.342 0.541 0.339 0.684 0.511 ...
 $ uti_max_credit_line           : num  0.514 0.418 0.342 0.543 0.633 ...
 $ pct_card_over_50_uti          : num  0.551 NA 0.451 0.608 0.574 ...
 $ ind_XYZ                       : int  0 0 0 0 0 0 0 1 0 1 ...
 $ rep_income                    : num  118266 89365 201365 191794 161465 ...
 $ rep_education                 : chr  "college" "college" "college" "college" ...
 $ Def_ind                       : int  0 0 0 0 0 0 0 0 0 0 ...
```

Figure 1. Visualize the dataset initial values to get a good approximation of the type of each variable.

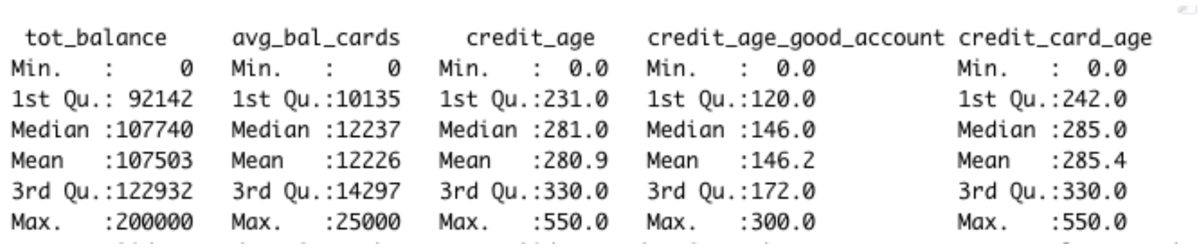
The initial values in the dataset as seen in Figure 1 give us an idea of type of each variable and some indication of which variable may be treated as a continuous variable as opposed to a categorical variable.

Continuous Variables

Continuous variables are those which are not divided into discrete values and preserve loss of data in numerical comparisons. Thus, some variables that are only integer values and within a range should still be treated as continuous variables for they preserve numerical comparisons among themselves. We consider the following variables as continuous in our dataset:

- tot_balance
- avg_bal_cards
- credit_age
- credit_age_good_account
- credit_card_age
- num_acc_30d_past_due_12_months
- num_acc_30d_past_due_6_months
- tot_amount_currently_past_due
- num_inq_12_month
- num_card_inq_24_month
- uti_open_card
- pct_over_50_uti
- uti_max_credit_line
- pct_card_over_50_uti
- rep_income
- num_card_12_month
- num_auto_.36_month

Now, let consider the summary statistics of these continuous variables. For our convenience let us look that the summaries of the first 5 continuous variables obtained using the summary() function.



```
  tot_balance      avg_bal_cards      credit_age     credit_age_good_account credit_card_age
 Min.   :     0   Min.   :    0   Min.   :  0.0   Min.   :  0.0           Min.   :  0.0
 1st Qu.: 92142   1st Qu.:10135   1st Qu.:231.0   1st Qu.:120.0           1st Qu.:242.0
 Median :107740   Median :12237   Median :281.0   Median :146.0           Median :285.0
 Mean   :107503   Mean   :12226   Mean   :280.9   Mean   :146.2           Mean   :285.4
 3rd Qu.:122932   3rd Qu.:14297   3rd Qu.:330.0   3rd Qu.:172.0           3rd Qu.:330.0
 Max.   :200000   Max.   :25000   Max.   :550.0   Max.   :300.0           Max.   :550.0
```

Figure 2. Summaries of the first 5 continuous variables

The first 5 continuous variables display some very interesting features. The substantial feature is the lack of a common scale among the variables. The variables are distributed on differing scales and this requires the need for normalization before model analysis can eb done on this data. Normalization represents data centred at the mean and in scale of the standard deviation which makes the data comparable. Normalization can be achieved by subtracting each value in a column by its mean and then dividing that whole term by its standard deviation. Other prominent feature observed among the first 5 variables is the concentration of data between the 1st and 3rd quarter, which may indicate the existence of normal distribution. However, this is hypothesis is only speculative and not indicative of all the continuous variables.

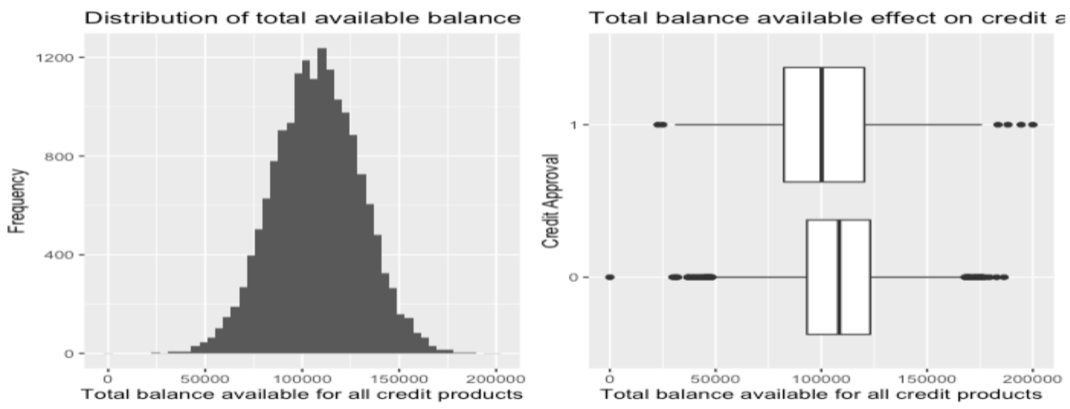


Figure 3. Histogram and boxplot analysis of tot_balance.

We can observe from Figure 3 that the distribution of Total balance available for credit approval is indicative of normal distribution with the mean of approximately 110000. Further, the boxplot shows that there is very little difference between credit approval or denial based solely on total balance with their means almost coinciding. This indicates the requirement of other variables to make better decision.

Next, we consider the Annual self-reported income and utilization on open credits to draw a connection to credit approval. These two factors seem to be important in the calculation of Credit since utilization dictates how much you spend and income defines your spending potential.

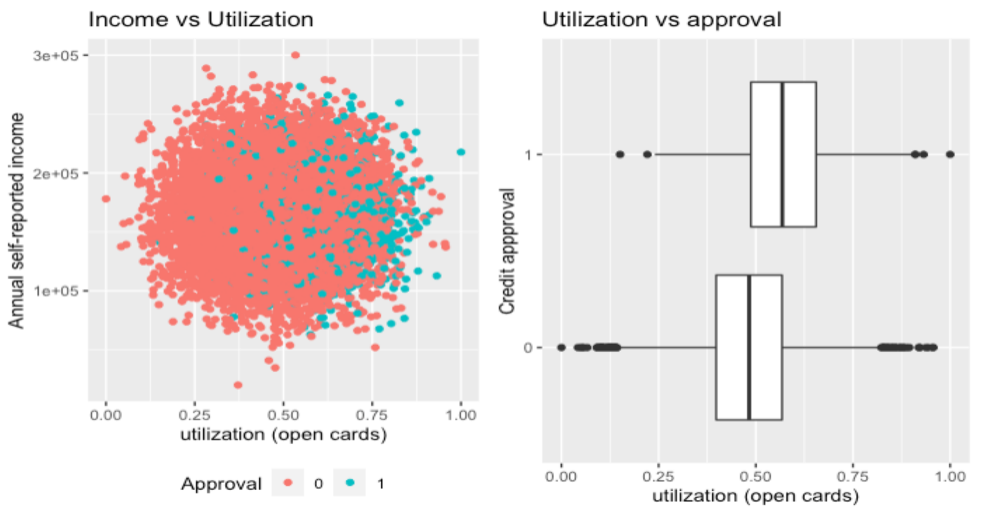


Figure 4. Income vs utilization trends and utilization effect on credit approval.

Observations from Figure 4 indicate that there is not much effect of annual income on credit approval. However, it can be noticed that approval usually tends to be in the higher utilization regions. This hypothesis is confirmed by the utilization vs approval boxplot indicating a slightly higher utilization for credit approval. Despite this trend, we cannot estimate the significance of it without an actual analysis model.

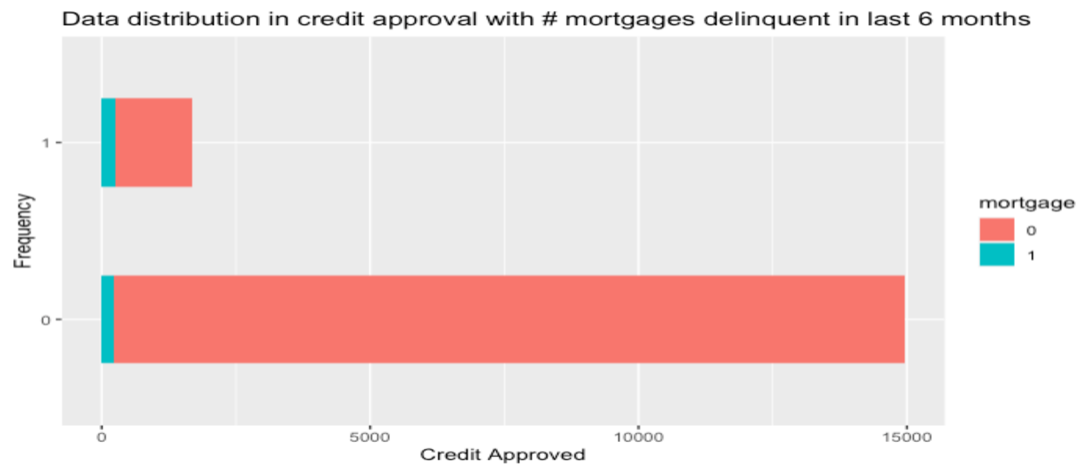


Figure 8. Distribution of credit approval over mortgages delinquent in the last 6 months

Next, as observed in Figure 8, there is a relationship between credit approval and whether any mortgages were delinquent in the last 6 months. The number of credit denials are overwhelmingly for no mortgage delinquency. The extent of this relationship cannot be determined be until model analysis, however, a relationship can be established without any doubt.

To get a better estimate of any trends or clusters in the dataset, we perform the PCA analysis to determine the direction of the descending variability. These descending direction of variability are an important analysis tool as they help us model the dataset and identify clusters without the computational requirement of all the data. By projecting the data along these orthogonal directions of maximal variability (principal components) we can reduce the dimension of the analysis and save expensive computing power and time. We first find the directions of descending variabilities and their associated standard deviations using the princomp() function, since there are more observations than variables. As mentioned previously we us normalized data vectors to ensure comparability. Once we have calculated the directions, we calculate the proportion of variance explained by each principal component to determine the number of components required to achieve good model replicability with less dimensions.

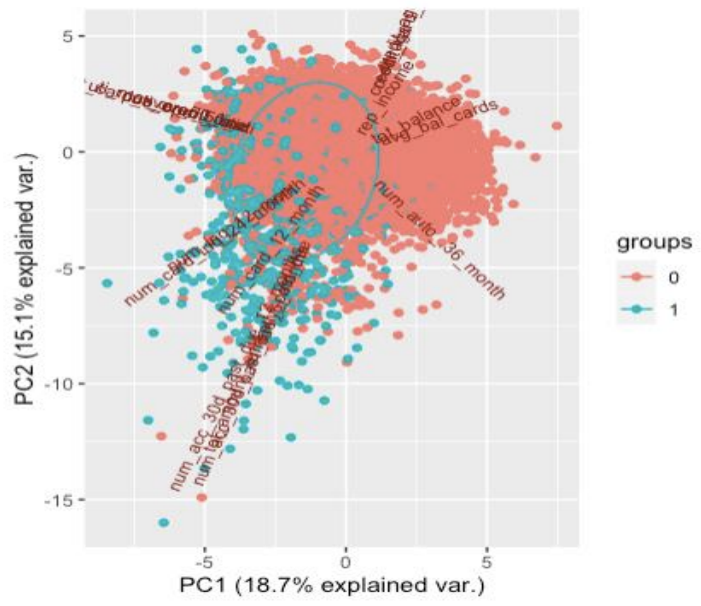


Figure 5. PCA biplot with first two components

The PCA biplot in Figure 5 shows the beginnings of two significant clusters when the data is projected along the first two principal components. These are encouraging signs of model success since only a total of only 38.3% (21.2% + 17.1%) of total variability of data in the first two principal components can help us distinguish two emerging clusters. However, this hypothesis must not be undertaken as proof since the dataset is heavily incline towards credit denial and thus the PCA biplot could be misleading. Next, we look onto the screen plot generated from the cumulated proportion of variance explained by adding one principal component at each iteration. As expected the proportion of variance explained by each PC decreases at each step, as seen by the decreasing slope of the screen plot. It can also be observed that with only 10 PC's about 90% of the data variability can be explained, while 12 PC's can explain over 98% of data variability. This is a significant dimension reduction from the original 17 dimensions.

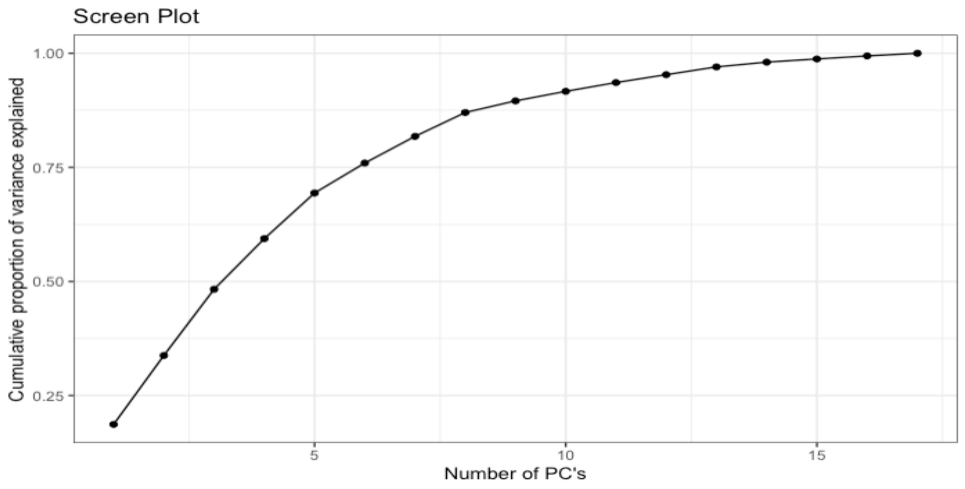


Figure 6. Screen plot with cumulative proportion of variance explained

As an addition it must be noted that this choice of continuous and categorical variables gives us the best screen plot, maximising the proportion of variance explained with the least number of dimensions, and hence this permutation choice of variables is being used for the model analysis in this final project.

Categorical Variables

A categorical variable can take one of a limited, and usually fixed, number of possible values, assigning each observation to a group or nominal category based on some qualitative property. Categorical variables have no quantitate property that can be compared to another property. Rather, it only serves to divide up the observations into different categories and dummy variables for better model performance and simplicity. The categorical variables for this model analysis are as follows:

- num_mortgage_currently_past_due
- ind_XYZ
- rep_education
- Def_ind

Let us try to visualize the different categories our dataset is divided into.

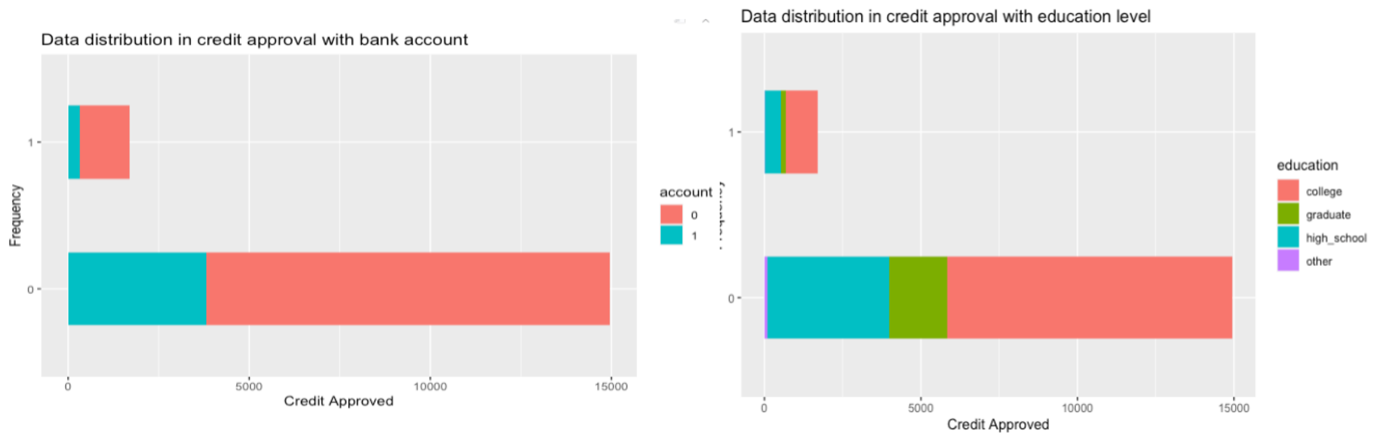


Figure 7. Distribution of credit approval over existing account in bank & education level

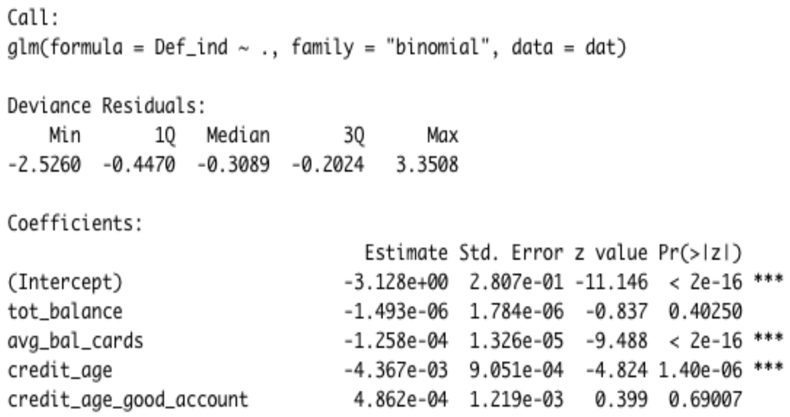
First, we look at the distribution of credit approvals against the variable determining an existing account in the bank in Figure 7. We see no general trends of credit approvals against existing accounts with both approval and denial being equally distributed between accounts. Similarly, we find no evidence of any general trends of credit being approved or denied based on the education level of the applicant. All education levels seem to be evenly distributed among the approved and denied observations and hence the null hypothesis is predicted to hold between these variables. It is important to note that this hypothesis has no concrete proof and hence should not be accepted as fact until model analysis.

Logistic Regression (Part 2)

Logistic regression is a predictor analysis model that is helpful is predicting categorical variables with a certain confidence. Logistic regression models the data on a S-curve, as opposed to a straight line in Linear Regression, and thus ensures the probability values to be between 0 and 1. This makes probability comparison easy and helps us better interpret the model.

We first begin by identifying the predictor and the response variables. In this dataset the variable '***Def_ind***' is considered to be the response while all the other variables are the predictors for this response.

Next, we begin with the simplest logistic regression model we can build where we include all the predictors as linear terms in the model. We use the glm() function to build a logistic regression model with the formula '*Def_ind ~ .*' and the *binomial family.*



```
Call:
glm(formula = Def_ind ~ ., family = "binomial", data = dat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5260  -0.4470  -0.3089  -0.2024   3.3508

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -3.128e+00  2.807e-01 -11.146  < 2e-16 ***
tot_balance             -1.493e-06  1.784e-06  -0.837  0.40250
avg_bal_cards           -1.258e-04  1.326e-05  -9.488  < 2e-16 ***
credit_age              -4.367e-03  9.051e-04  -4.824 1.40e-06 ***
credit_age_good_account  4.862e-04  1.219e-03   0.399  0.69007
```

Figure 9. glm() call example

As observed from the head of the glm() call in Figure 9, it echoes the formula used and then the deviance residuals for the model. It also provides us with the P-values(>|z|) which indicates the significance of the estimated coefficient in the model (where z is the normalized position on a standard normal curve). The standard error values represent the variability in the data and how much it moves around the centred mean. At the bottom of the glm() call we are provided with the AIC value for the current model. This value is significant in determining the best model and is known as the stepAIC method with the goal of minimizing this value.

Another common method to check a model against another model is to use the ROC curve, which evaluates the model and many different classification threshold values. The area under the curve (AUC) is an indicator of goodness of model fit. The advantage of this method over prediction accuracy is that it is scale-invariant as it measures how well predictions are ranked, and classification-threshold-invariant as it measures the quality of the method's predictions. Figure 10 shows the ROC curve with the AUC value at the bottom for the current logistic regression model.

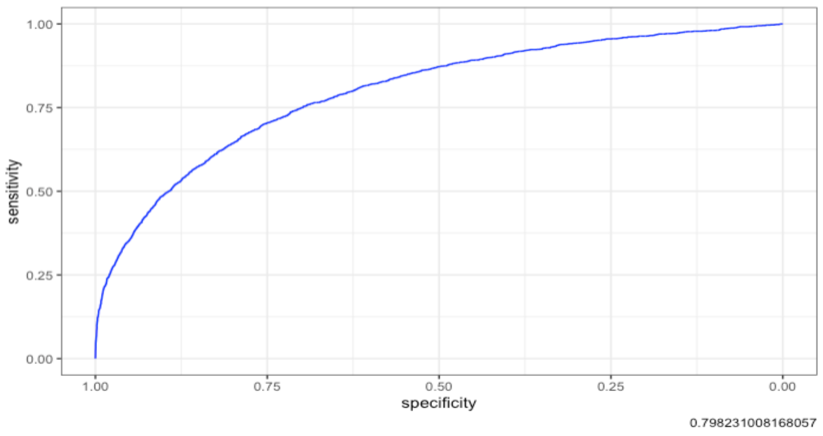


Figure 10. ROC curve for simple logistic regression model with AUC value at the bottom.

We will use this ROC curve as the benchmark for further models, trying to increase the AUC from on here. (AUC_{curr} = 0.798)

(Note: We also check the data for any collinearity. However, no collinear terms were found using the VIF factor so we do not include it in the code or the model analysis.)

Next, we use the stepAIC() method to determine the regression model with the lowest AIC value, which we estimate to be a better fit that the current model. The ROC curve for the returned model is as follows.

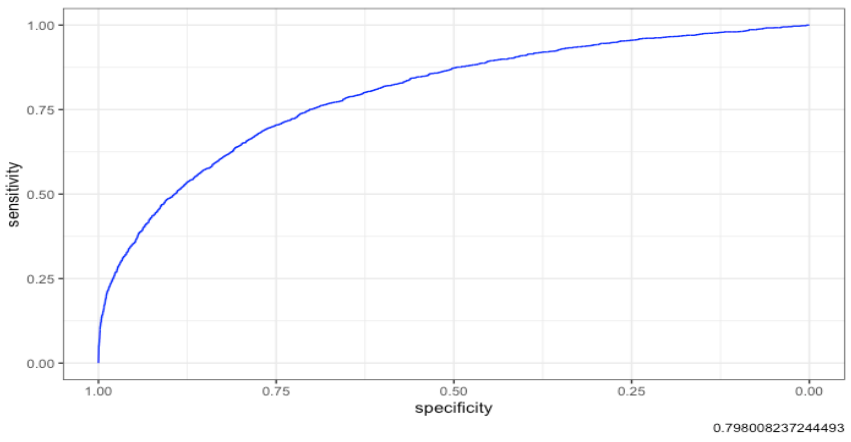


Figure 11. ROC curve for lowest AIC value. AUC = 0.798. Thus, it does not improve on the current model.

As observed, there is no significant increase in the AUC value (=0.798) and thus we retain our AUC_{curr} model.

Now that we have tested all the linear terms, we venture further and add interaction terms to the model to find a better fit. Interaction terms include interaction between the continuous and categorical variables which add to the formula in a meaningful way terms. These interaction terms act as moderators on the final response, with categorical variables moderating continuous variables. It is quite difficult to establish these interactions just by looking at the data. Thus, we use a brute-force approach by checking every interaction term and use the stepAIC method to calculate the model with the lowest AIC.

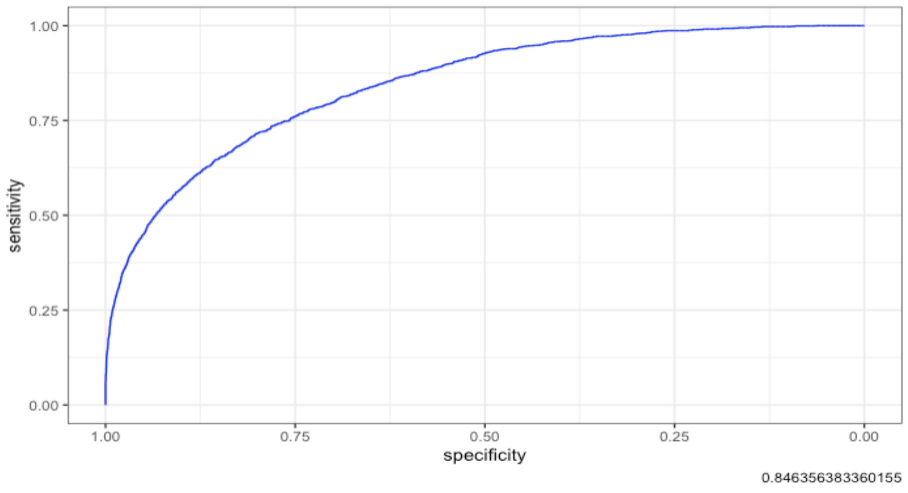


Figure 12. ROC curve with interaction terms with lowest AIC value. AUC = 0.8464

As observed, the model formula has 13 interaction terms that moderate the continuous variables to the response to build a better model. This model can be deemed better based on the AUC value of the ROC curve. Thus, we adopt this model as our current best model with the AUC_{curr} = 0.8464.

As an exercise we now try to predict the accuracy of the current model. The accuracy of the model on the test data is found to be about 92%.

Next we try to add non-linear terms to find a better fit for the model. To save computation power, we manually check each interaction term up to the 3rd power to maximise the AUC value, while still maintaining model simplicity. This is necessary to model these non-linear terms for a better fit with more flexibility in the data. We find that the non-linear terms that maximise model simplicity are tot_balance^2 and avg_bal_cards^3 added to the current model. Thus, the final logistic regression model has the formula as follows:

```
formula = Def_ind ~ tot_balance + num_acc_30d_past_due_12_months + uti_open_card +
num_inq_12_month + credit_age + avg_bal_cards + num_acc_30d_past_due_6_months +
ind_XYZ + num_card_12_month + num_mortgage_currently_past_due + pct_over_50_uti +
num_card_inq_24_month + tot_amount_currently_past_due + rep_education +
```

num_acc_30d_past_due_12_months:num_inq_12_month + tot_balance:avg_bal_cards + uti_open_card:ind_XYZ + avg_bal_cards:num_card_12_month + _open_card:num_inq_12_month + tot_balance:num_acc_30d_past_due_6_months + tot_balance:credit_age + uti_open_card:num_card_inq_24_month + num_inq_12_month:avg_bal_cards + num_card_12_month:rep_education + _inq_12_month:num_mortgage_currently_past_due + credit_age:num_card_12_month + tot_balance:num_card_12_month + I(tot_balance^2) + I(avg_bal_cards^3)

The corresponding ROC curve is:

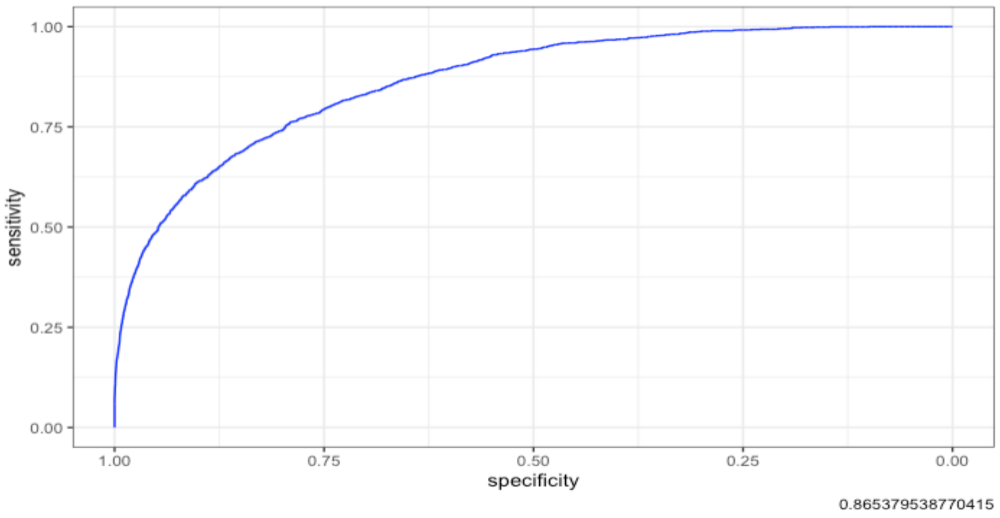


Figure 13. ROC curve for final model. AUC = 0.8654

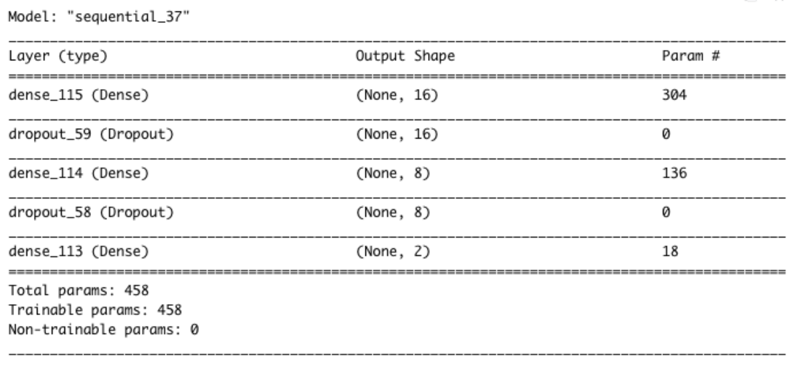
We see that we have maximized the AUC (=0.8654) and thus this is the best model fit. We now try to calculate the accuracy of this model for comparison with other models. The accuracy of the model is found to be around 91.1% for a classification threshold of 0.5. This accuracy is not a significant improvement in the previous accuracy, however this model is better equipped to handle varying classification thresholds and thus is the preferred choice.

Neural Network (Part 3)

Neural networks are networks or circuits of neurons composed of artificial neurons or nodes. The connections of the biological neurons are modelled as weights. It uses reinforcement as positive and negative weights to guide the model towards the optimal accuracy. Neural networks are powerful as they can identify patterns and make relationships in a seemingly unrelated dataset.

To run a neural network through the dataset we first have to pre-process the data. To pre-process the data I removed the categorical variables of rep_education as it requires an intermittent embedded layer which is beyond the scope of this project. Next I divided the training and testing dataset into response and predictor sets with the response being a categorical variable and the predictors as numeric variables.

Next, we begin to build the sequential neural-network model and adjusting its parameters. We build a 1-hidden-layer neural network with 16 neurons in the first layer and 8 neurons in the second layer. We find that these parameters minimize loss and eliminates overfitting of data. Between each layer there is a drop layer with drop rate of 0.1.



```
Model: "sequential_37"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_115 (Dense)            (None, 16)                304
_________________________________________________________________
dropout_59 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_114 (Dense)            (None, 8)                 136
_________________________________________________________________
dropout_58 (Dropout)         (None, 8)                 0
_________________________________________________________________
dense_113 (Dense)            (None, 2)                 18
=================================================================
Total params: 458
Trainable params: 458
Non-trainable params: 0
_________________________________________________________________
```

Figure 14. The neural network model layers and parameters.

Next, we train the neural network on out dataset. The loss is compiled using '*categorical crossentropy*' method while '*adam's*' optimizer was used to optimize and calculate the step variable. Being a logistic classifier the metric choice was taken as '*accuracy*'. We set the epoch and batch-size as 15 to balance computational power and time.

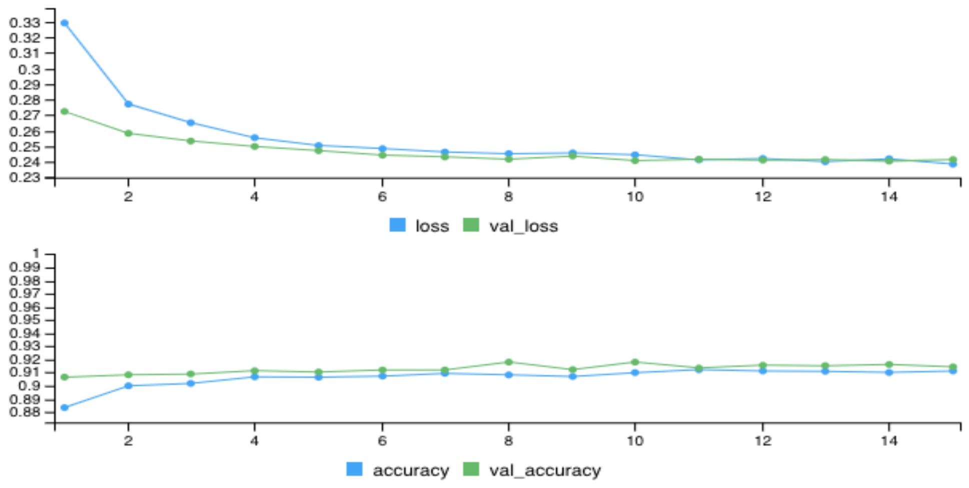


Figure 15. Loss and accuracy of the trained model at each epoch

As observed in Figure 15, the loss function was minimized at approximately 0.24 and the accuracy peaked at approximately 91%. These statistics do not indicate an exceptional neural-network, however, it does represent the peak value for a various parameter adjustments.

Finally, we test this neural network on out test data. It outputs an expected accuracy value of 91.4% over the testing data.

Which model is better and/or preferred?

We have developed extensive logistic regression and neural network models to predict credit approval given the historical dataset. The performance of both models was found to be comparable, with an average accuracy of 91.2%. This is an encouraging sign since despite two completely different approaches we could achieve the same accuracy and so there is validation in the methodologies. Further, after careful consideration, the preferred model for this dataset is the logistic regression model. With comparable performance, we choose the model with less complexity and one which requires less computation power. Logistic regression is easier to develop and tweak than a neural-network with hidden layers. Furthermore, it is not imprudent to think that logistic regression when projected along the principal components would yield better results with even less computation power and dimensions. Simplicity in the model is the deciding criteria for this dataset.

Pros and Cons

Logistic Regression

- Pros: Manageable number of predictor variables, No significant collinearity, categorical variable with only 2 levels as response, significant interaction terms that could be modelled using stepAIC method
- Cons: Time consuming to build the model and requires significant computational power to run the stepAIC method, ROC curve values change only marginally so trade-off between simplicity and model-fit hard to judge, lopsided dataset with significant difference in number of response terms.

Neural Network:

- Pros: Easy to build the model using keras package, small dataset requires less neurons and hence less computational power, binary prediction results in better accuracy.

- Cons: Substantial loss in the dataset, hard to model categorical variables without embedded layer, accuracy almost same as logistic regression despite more computation and complexity.

Final Project

Aman Kothari

05/05/2021

Initial Set-up

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.1.0     ✓ dplyr   1.0.4
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Part 1: Data Visualization

```
dat = read.csv("../Case-study-training-data.csv")
dat_test = read.csv("../Case-study-test-data.csv")
dat_test = tibble(dat_test)
dat = tibble(dat)
str(dat)
```

```
## tibble[,21] [20,000 × 21] (S3: tbl_df/tbl/data.frame)
##  $ tot_balance                   : num [1:20000] 102956 132759 124659 133969 143602
...
##  $ avg_bal_cards                 : num [1:20000] 14819 18952 15348 14051 14859 ...
##  $ credit_age                    : int [1:20000] 238 384 277 375 374 250 249 252 263 3
28 ...
##  $ credit_age_good_account       : int [1:20000] 104 197 110 224 155 178 132 139 102 1
69 ...
##  $ credit_card_age               : int [1:20000] 264 371 288 343 278 255 251 269 269 3
28 ...
##  $ num_acc_30d_past_due_12_months : int [1:20000] 0 0 0 0 0 1 0 0 0 0 ...
##  $ num_acc_30d_past_due_6_months  : int [1:20000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ num_mortgage_currently_past_due: int [1:20000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tot_amount_currently_past_due  : num [1:20000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ num_inq_12_month              : int [1:20000] 0 0 0 2 0 0 0 0 0 0 ...
##  $ num_card_inq_24_month         : int [1:20000] 0 0 0 2 0 1 0 0 0 0 ...
##  $ num_card_12_month             : int [1:20000] 1 0 0 1 0 0 0 0 0 0 ...
##  $ num_auto_.36_month            : int [1:20000] 0 0 0 0 0 0 0 0 1 0 ...
##  $ uti_open_card                 : num [1:20000] 0.367 0.491 0.359 0.7 0.647 ...
##  $ pct_over_50_uti               : num [1:20000] 0.342 0.541 0.339 0.684 0.511 ...
##  $ uti_max_credit_line           : num [1:20000] 0.514 0.418 0.342 0.543 0.633 ...
##  $ pct_card_over_50_uti          : num [1:20000] 0.551 NA 0.451 0.608 0.574 ...
##  $ ind_XYZ                       : int [1:20000] 0 0 0 0 0 0 0 1 0 1 ...
##  $ rep_income                    : num [1:20000] 118266 89365 201365 191794 161465 ...
##  $ rep_education                 : chr [1:20000] "college" "college" "college" "colleg
e" ...
##  $ Def_ind                       : int [1:20000] 0 0 0 0 0 0 0 0 0 0 ...
```

```
dat = na.omit(dat)
dat$Def_ind = as.factor(dat$Def_ind)
dat$ind_XYZ = as.factor(dat$ind_XYZ)
dat$rep_education = as.factor(dat$rep_education)
levels(dat$rep_education)
```

```
## [1] "college"     "graduate"    "high_school" "other"
```

```
dat_test$Def_ind = as.factor(dat_test$Def_ind)
dat_test$ind_XYZ = as.factor(dat_test$ind_XYZ)
dat_test = dat_test[!dat_test$rep_education == "",]
dat_test$rep_education = as.factor(dat_test$rep_education)
#dat$num_mortgage_currently_past_due = as.factor(dat$num_mortgage_currently_past_due)
#dat$num_card_12_month = as.factor(dat$num_card_12_month)
#dat$num_auto_.36_month = as.factor(dat$num_auto_.36_month)
#dat$num_acc_30d_past_due_6_months = as.factor(dat$num_acc_30d_past_due_6_months)
summary(dat)
```

```
##   tot_balance     avg_bal_cards     credit_age    credit_age_good_account
##  Min.   :     0   Min.   :    0   Min.   :  0.0   Min.   :  0.0
##  1st Qu.: 92142   1st Qu.:10135   1st Qu.:231.0   1st Qu.:120.0
##  Median :107740   Median :12237   Median :281.0   Median :146.0
##  Mean   :107503   Mean   :12226   Mean   :280.9   Mean   :146.2
##  3rd Qu.:122932   3rd Qu.:14297   3rd Qu.:330.0   3rd Qu.:172.0
##  Max.   :200000   Max.   :25000   Max.   :550.0   Max.   :300.0
##  credit_card_age num_acc_30d_past_due_12_months num_acc_30d_past_due_6_months
##  Min.   :  0.0   Min.   :0.0000                 Min.   :0.00000
##  1st Qu.:242.0   1st Qu.:0.0000                 1st Qu.:0.00000
##  Median :285.0   Median :0.0000                 Median :0.00000
##  Mean   :285.4   Mean   :0.1579                 Mean   :0.02936
##  3rd Qu.:330.0   3rd Qu.:0.0000                 3rd Qu.:0.00000
##  Max.   :550.0   Max.   :5.0000                 Max.   :2.00000
##  num_mortgage_currently_past_due tot_amount_currently_past_due
##  Min.   :0.0000                  Min.   :    0.0
##  1st Qu.:0.0000                  1st Qu.:    0.0
##  Median :0.0000                  Median :    0.0
##  Mean   :0.0299                  Mean   :  354.2
##  3rd Qu.:0.0000                  3rd Qu.:    0.0
##  Max.   :1.0000                  Max.   :35000.0
##  num_inq_12_month  num_card_inq_24_month num_card_12_month num_auto_.36_month
##  Min.   : 0.0000   Min.   : 0.000        Min.   :0.0000    Min.   :0.000
##  1st Qu.: 0.0000   1st Qu.: 0.000        1st Qu.:0.0000    1st Qu.:0.000
##  Median : 0.0000   Median : 0.000        Median :0.0000    Median :0.000
##  Mean   : 0.6133   Mean   : 1.044        Mean   :0.2723    Mean   :0.165
##  3rd Qu.: 1.0000   3rd Qu.: 1.000        3rd Qu.:1.0000    3rd Qu.:0.000
##  Max.   :10.0000   Max.   :18.000        Max.   :3.0000    Max.   :2.000
##  uti_open_card    pct_over_50_uti  uti_max_credit_line pct_card_over_50_uti
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000      Min.   :0.0000
##  1st Qu.:0.4048   1st Qu.:0.4011   1st Qu.:0.3778      1st Qu.:0.4643
##  Median :0.4909   Median :0.4855   Median :0.4649      Median :0.5518
##  Mean   :0.4914   Mean   :0.4842   Mean   :0.4653      Mean   :0.5511
##  3rd Qu.:0.5783   3rd Qu.:0.5679   3rd Qu.:0.5541      3rd Qu.:0.6384
##  Max.   :1.0000   Max.   :0.9294   Max.   :1.0000      Max.   :1.0000
##  ind_XYZ     rep_income         rep_education   Def_ind
##  0:12512   Min.   : 20000   college    :10104   0:14956
##  1: 4141   1st Qu.:143751   graduate   : 2026   1: 1697
##            Median :166630   high_school: 4403
##            Mean   :166504   other      :  120
##            3rd Qu.:189020
##            Max.   :300000
```

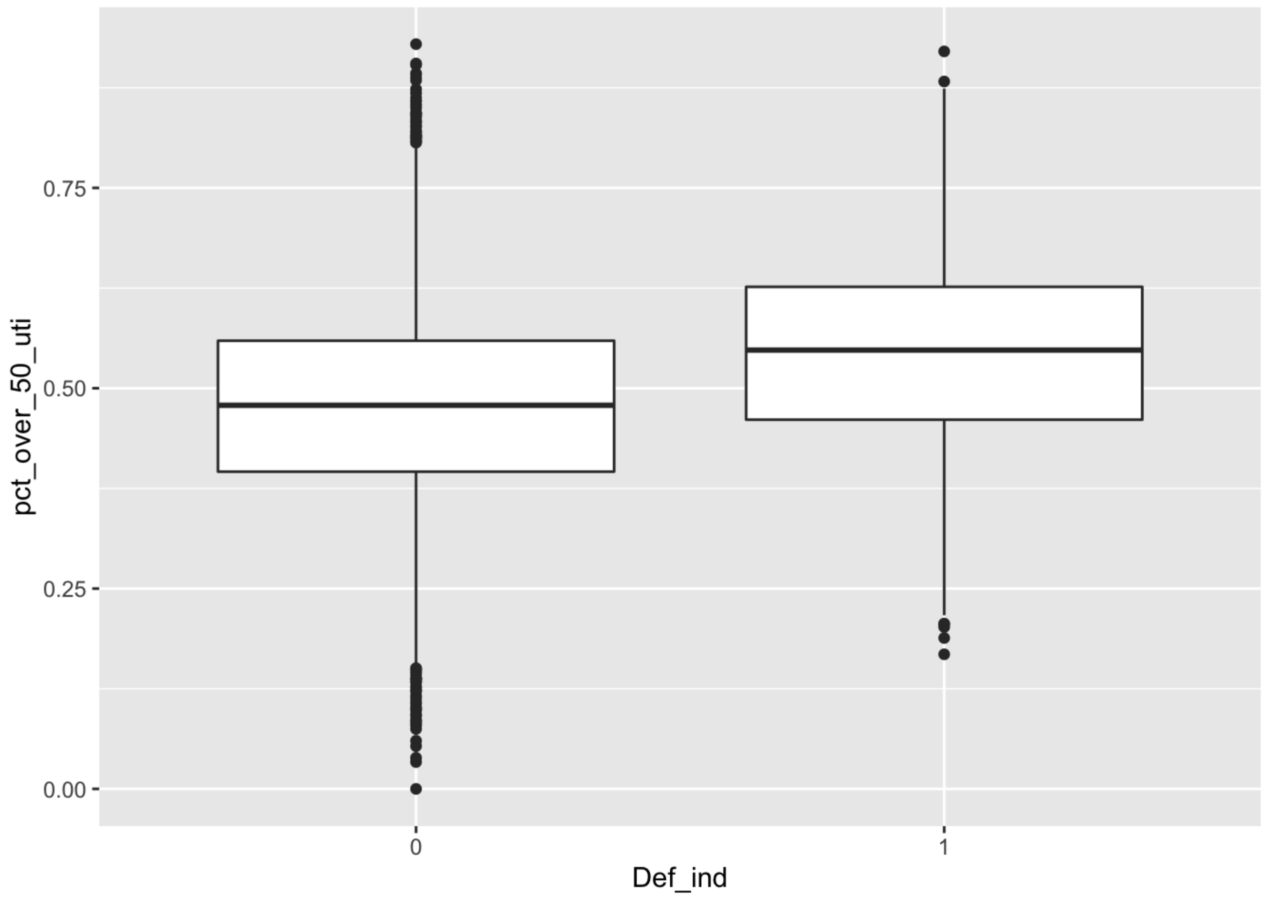
```
#p1 = ggplot(dat, aes(tot_balance)) + geom_histogram(bins = 50) + xlab("Total balance avai
lable for all credit products") + ylab("Frequency") + ggtitle("Distribution of total avail
able balance")

#p2 = ggplot(dat, aes(tot_balance, Def_ind)) + geom_boxplot() + labs(title = "Total balanc
e available effect on credit approval", x = "Total balance available for all credit produc
ts", y = "Credit Approval")

#plot_grid(p1, p2)

#ggplot(dat, aes(rep_income, Def_ind)) + geom_boxplot(notch = T) + labs(title = "Annual In
come effect on credit approval", x = "Annual Income (self-reported)", y = "Credit Approva
l")

#p1 = ggplot (dat, aes(uti_open_card,rep_income, color = Def_ind)) + geom_point() + labs(t
itle = "Income vs Utilization", x = "utilization (open cards)", y = "Annual self-reported
 income", color = "Approval") + theme(legend.position = "bottom")
#p2 = ggplot(dat, aes(uti_open_card, Def_ind)) + geom_boxplot() + labs(title = "Utilizatio
n vs approval", x = "utilization (open cards)", y = "Credit appproval")
#plot_grid(p1, p2)
ggplot(dat, aes(Def_ind, pct_over_50_uti)) + geom_boxplot()
```



```
con_dat = (dat[,c(-21, -20, -18, -8)])
dat.pca.scaled = princomp(con_dat, cor=T)
library(ggbiplot)
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
## Loading required package: scales
```

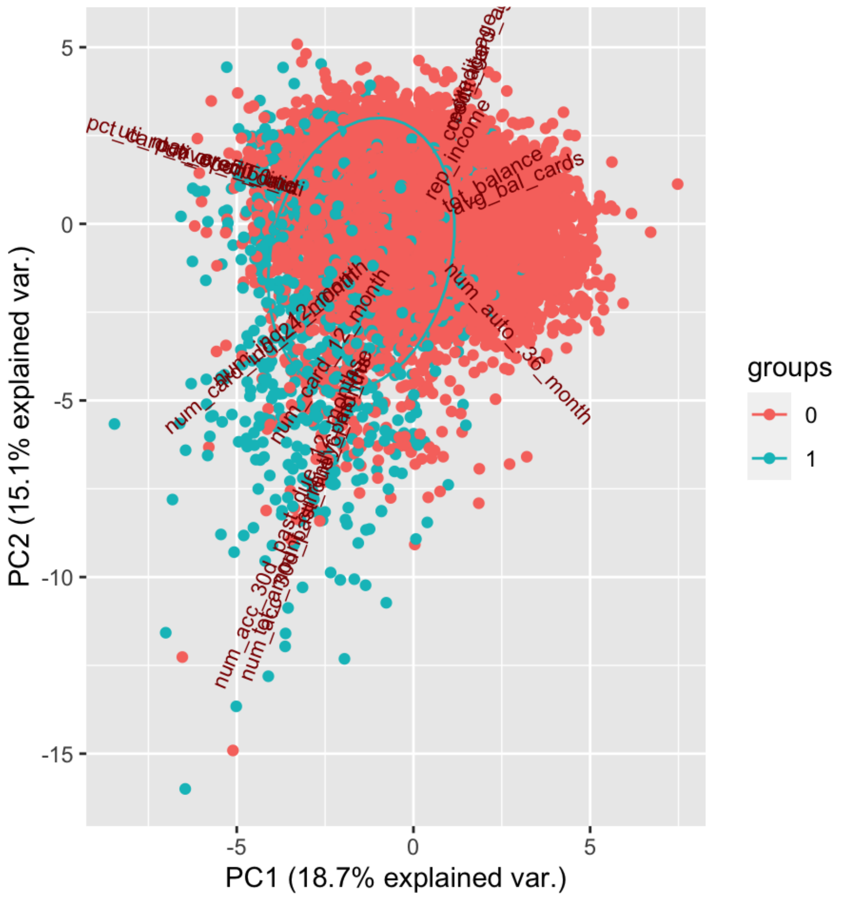
```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

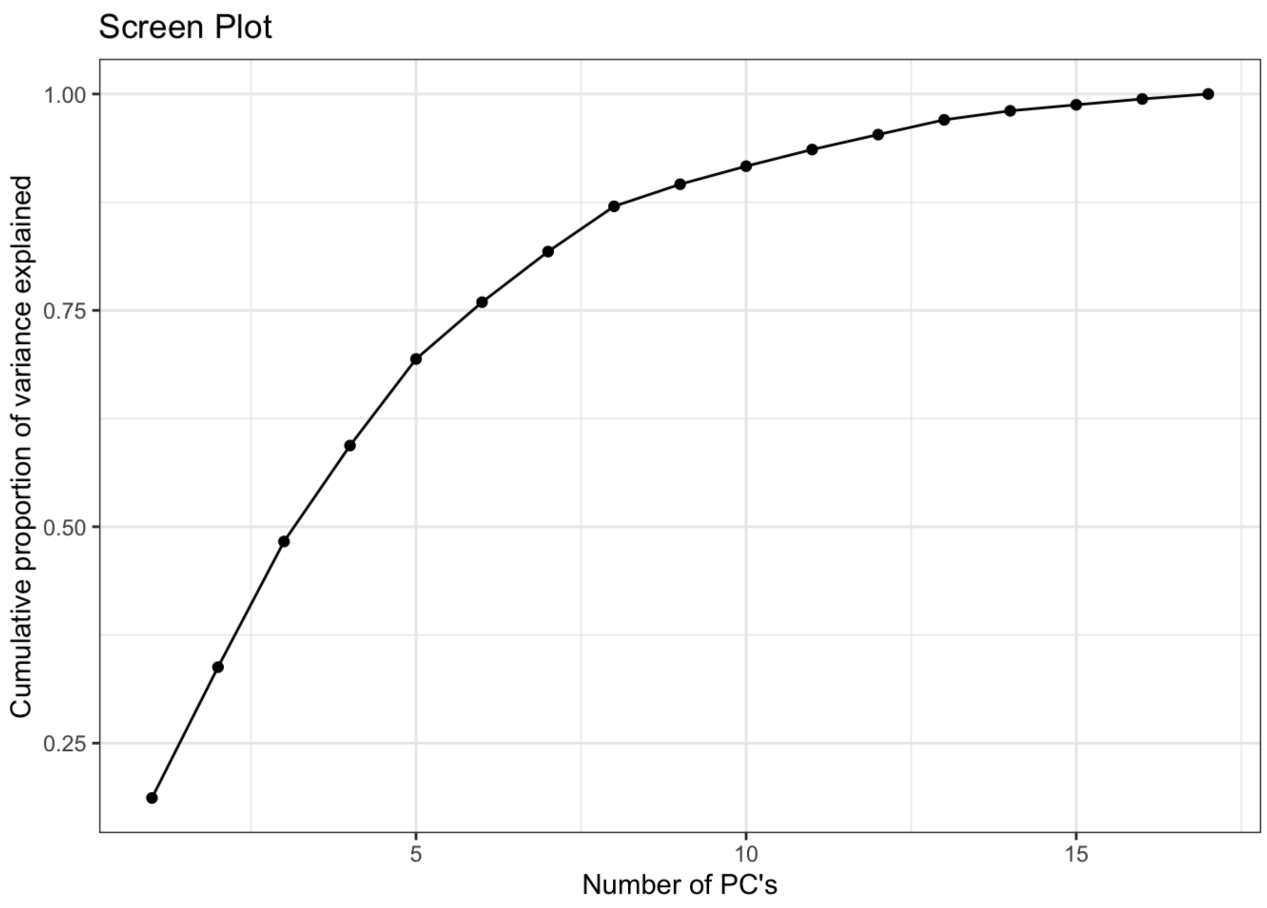
```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
## Loading required package: grid
```

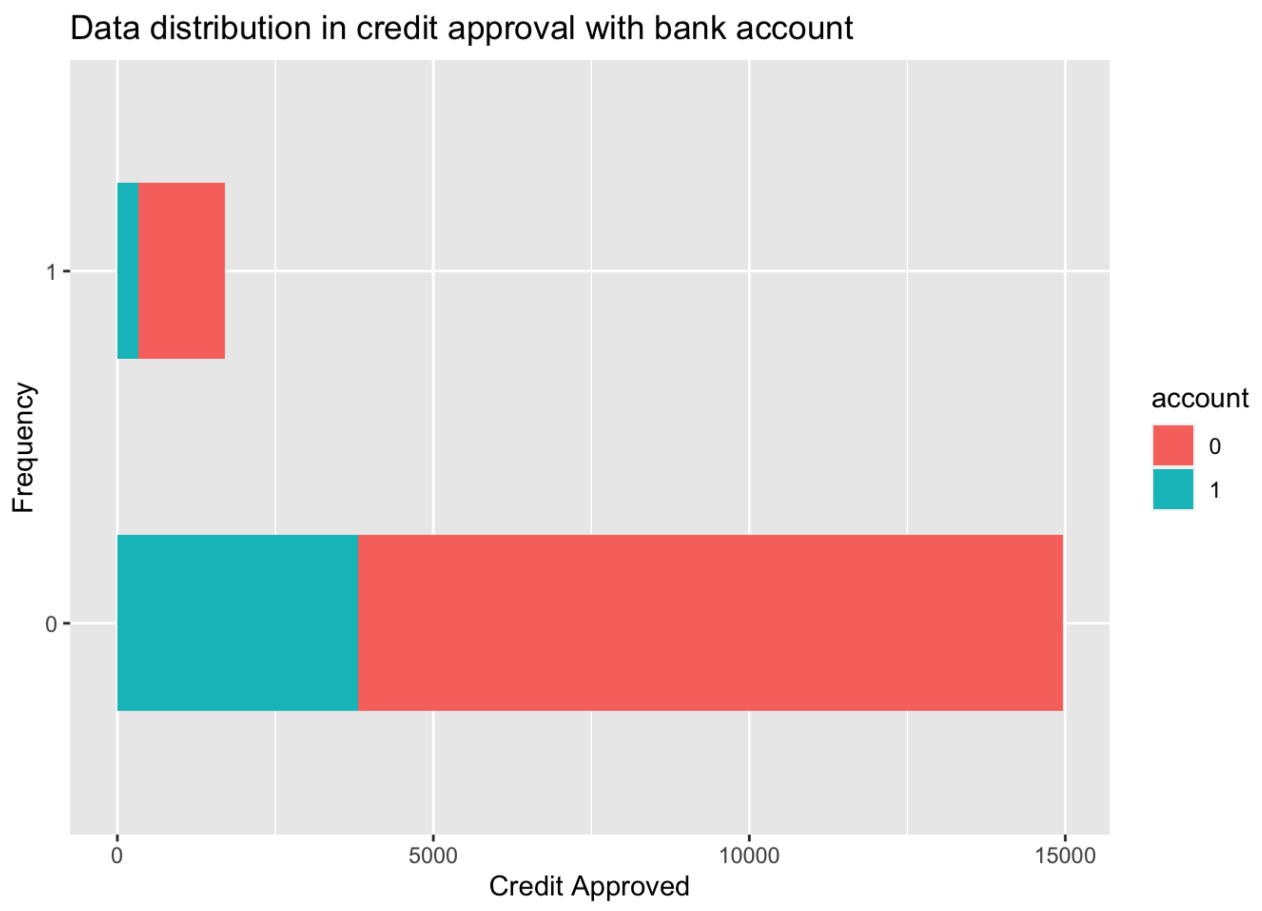
```
ggbiplot(dat.pca.scaled, groups =dat$Def_ind,  ellipse = TRUE, obs.scale = 1, var.scale =
1)
```



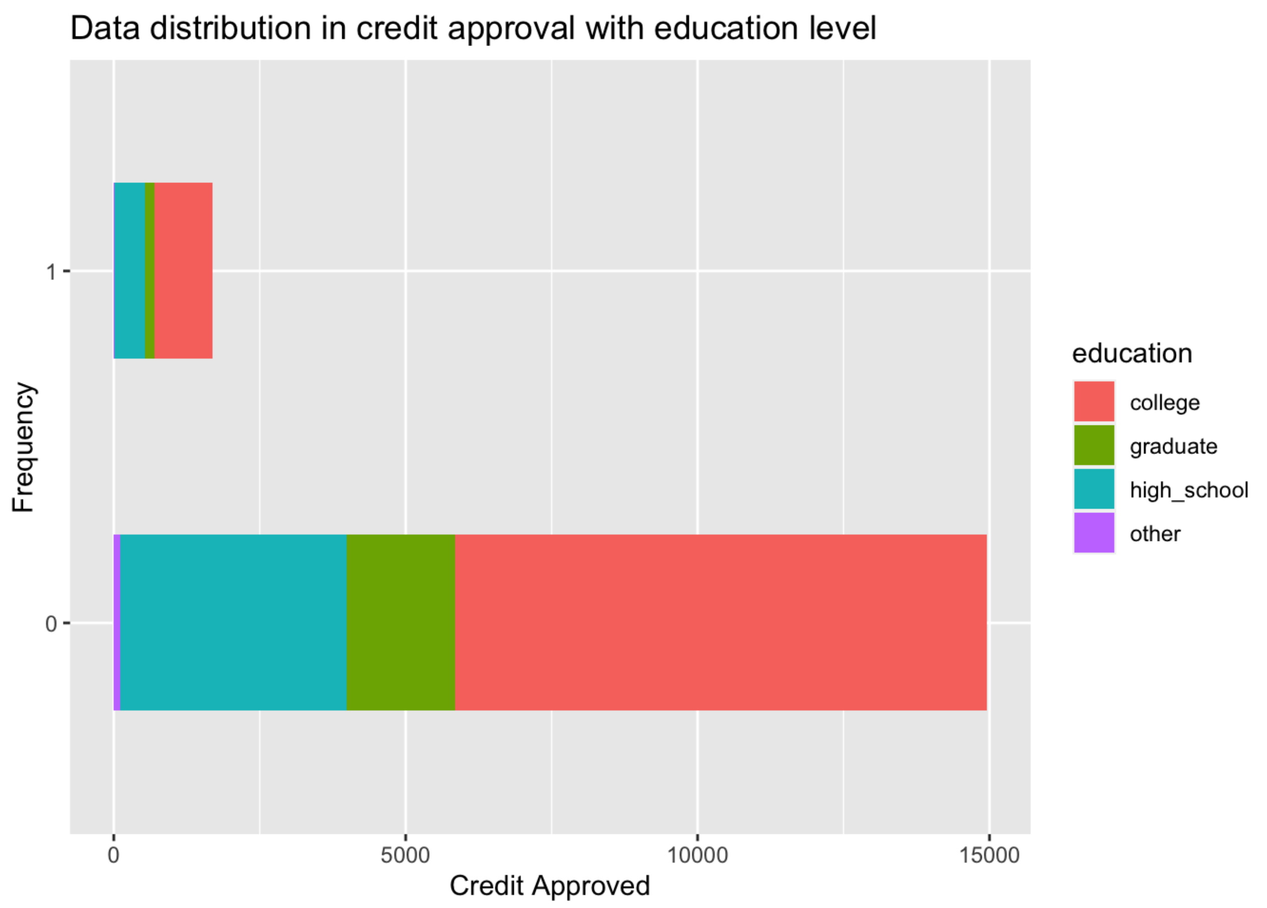
```
df = data.frame(npc = 1:ncol(con_dat), cpve = cumsum(dat.pca.scaled$sdev^2)/sum(dat.pca.sc
aled$sdev^2))
ggplot(df, aes(npc, cpve)) + geom_line() + geom_point() + theme_bw() + labs(title = "Scree
n Plot", x = "Number of PC's", y = "Cumulative proportion of variance explained")
```



```
ggplot(dat, aes(Def_ind, fill = ind_XYZ)) + geom_bar(width = 0.5) + coord_flip() + labs(ti
tle = "Data distribution in credit approval with bank account", x = "Frequency", y = "Cred
it Approved", fill = "account")
```



```
ggplot(dat, aes(Def_ind, fill = rep_education)) + geom_bar(width = 0.5) + coord_flip() + l
abs(title = "Data distribution in credit approval with education level", x = "Frequency",
 y = "Credit Approved", fill = "education")
```



```
##
```

Logistic Regression

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

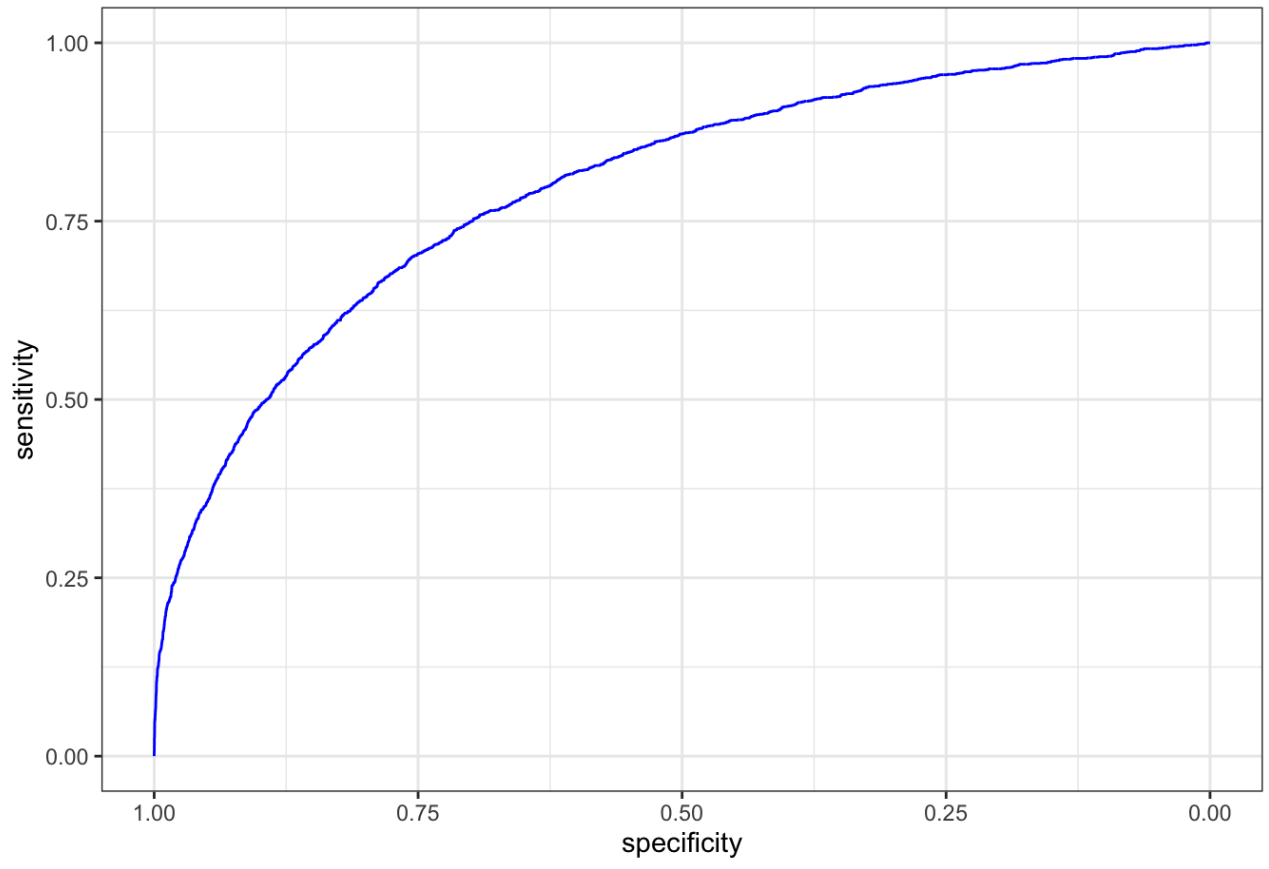
```
fit0 = glm(Def_ind~tot_balance, family = "binomial", data = dat)
fit1 = glm(Def_ind ~ . , family = "binomial", data = dat)
summary(fit1)
```

```
##
## Call:
## glm(formula = Def_ind ~ ., family = "binomial", data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5260  -0.4470  -0.3089  -0.2024   3.3508
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -3.128e+00  2.807e-01 -11.146  < 2e-16 ***
## tot_balance                    -1.493e-06  1.784e-06  -0.837  0.40250
## avg_bal_cards                  -1.258e-04  1.326e-05  -9.488  < 2e-16 ***
## credit_age                     -4.367e-03  9.051e-04  -4.824 1.40e-06 ***
## credit_age_good_account         4.862e-04  1.219e-03   0.399  0.69007
## credit_card_age                -3.697e-04  8.350e-04  -0.443  0.65793
## num_acc_30d_past_due_12_months  9.309e-01  8.239e-02  11.299  < 2e-16 ***
## num_acc_30d_past_due_6_months   4.353e-01  1.778e-01   2.448  0.01436 *
## num_mortgage_currently_past_due 3.193e-01  1.890e-01   1.689  0.09124 .
## tot_amount_currently_past_due   1.018e-05  2.232e-05   0.456  0.64818
## num_inq_12_month                3.590e-01  5.160e-02   6.957 3.47e-12 ***
## num_card_inq_24_month          -4.934e-02  2.910e-02  -1.696  0.08998 .
## num_card_12_month               1.566e-01  5.576e-02   2.808  0.00499 **
## num_auto_.36_month              5.772e-02  7.431e-02   0.777  0.43733
## uti_open_card                   5.459e+00  5.484e-01   9.954  < 2e-16 ***
## pct_over_50_uti                 7.095e-01  3.483e-01   2.037  0.04165 *
## uti_max_credit_line             1.068e-01  3.283e-01   0.325  0.74492
## pct_card_over_50_uti           -3.000e-01  4.158e-01  -0.721  0.47071
## ind_XYZ1                       -2.865e-01  7.025e-02  -4.079 4.53e-05 ***
## rep_income                      8.152e-07  8.459e-07   0.964  0.33515
## rep_educationgraduate          -6.508e-02  9.487e-02  -0.686  0.49269
## rep_educationhigh_school        1.272e-01  6.369e-02   1.997  0.04584 *
## rep_educationother             -3.588e-01  3.725e-01  -0.963  0.33553
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10965.9  on 16652  degrees of freedom
## Residual deviance:  8815.9  on 16630  degrees of freedom
## AIC: 8861.9
##
## Number of Fisher Scoring iterations: 6
```

```
fit_roc = roc(dat$Def_ind, fit1$fitted.values, levels=c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
ggroc(fit_roc, color="blue") + theme_bw() + labs(caption = fit_roc$auc)
```



0.798231008168057

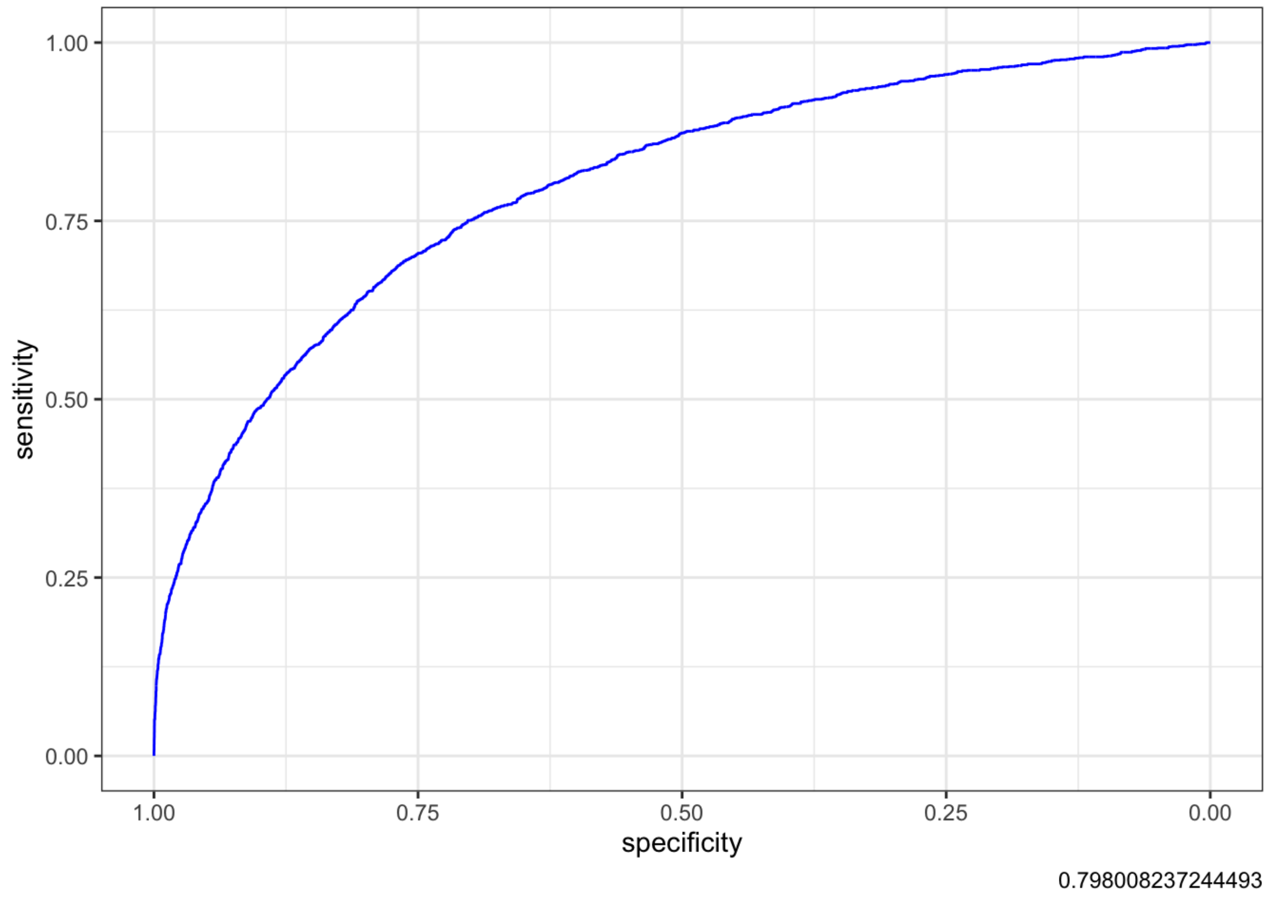
```
{ sink("/dev/null"); aic_both_l = stepAIC(fit0, direction= "both", scope = list(upper=fit
1, lower=fit0)); sink(); }
fit_roc = roc(dat$Def_ind, aic_both_l$fitted.values, levels=c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
fit_roc$auc
```

```
## Area under the curve: 0.798
```

```
ggroc(fit_roc, color="blue") + theme_bw() + labs(caption = fit_roc$auc)
```



```
fit0 = glm(Def_ind~tot_balance, family = "binomial", data = dat)
fit1 = glm(Def_ind ~ .^2 , family = "binomial", data = dat)
{ sink("/dev/null"); aic_both_1 = stepAIC(fit0, direction= "both", scope = list(upper=fit
1, lower=fit0)); sink(); }
aic_both_1$aic
```

```
## [1] 7989.541
```

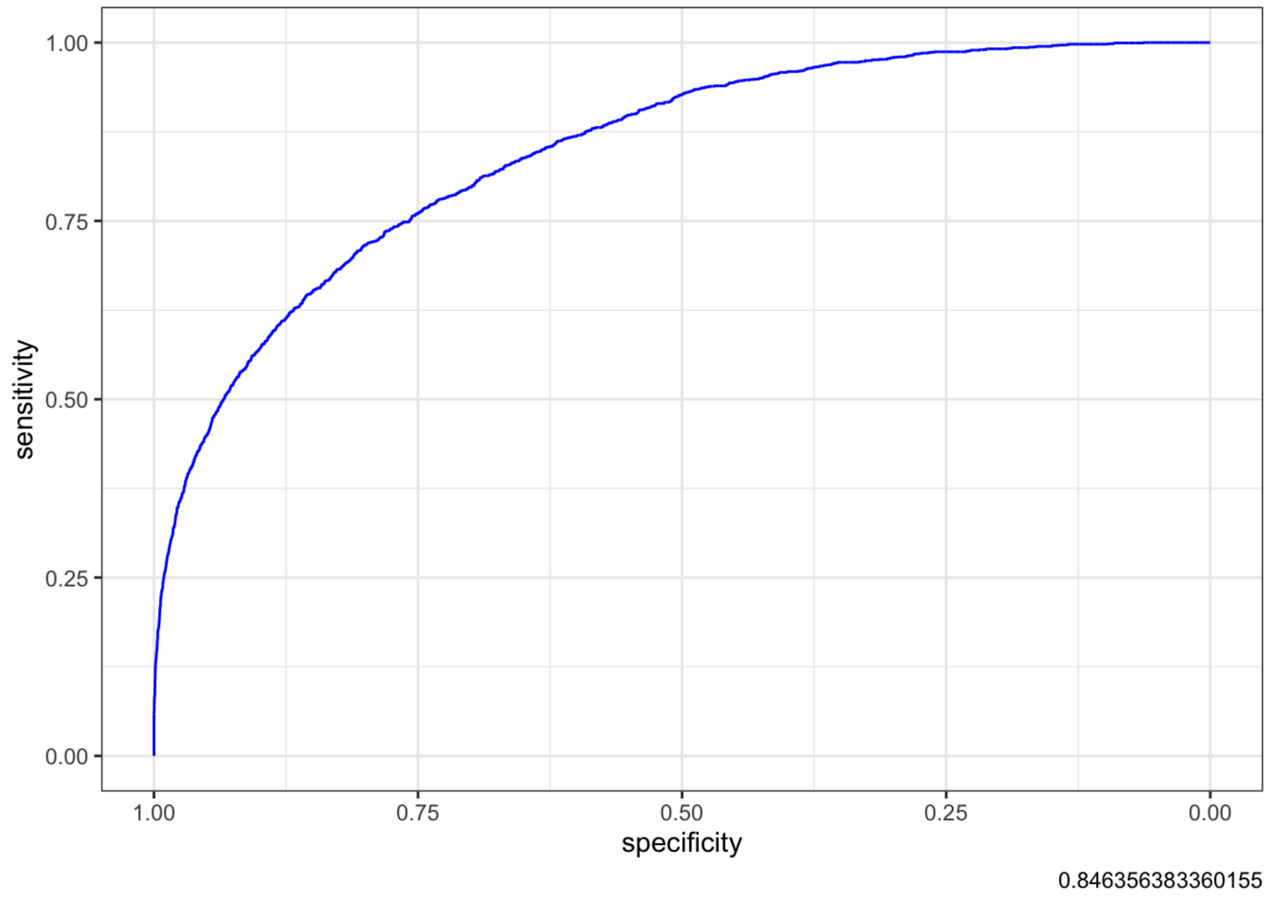
```
fit_roc = roc(dat$Def_ind, aic_both_1$fitted.values, levels=c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
fit_roc$auc
```

```
## Area under the curve: 0.8464
```

```
ggroc(fit_roc, color="blue") + theme_bw() + labs(caption = fit_roc$auc)
```



```
glm.probs =predict(aic_both_1, dat_test, type="response")
probs = ifelse(glm.probs > 0.5, 1, 0)
a <- table(probs)
accuracy = a[1] / (a[1]+a[2])
```

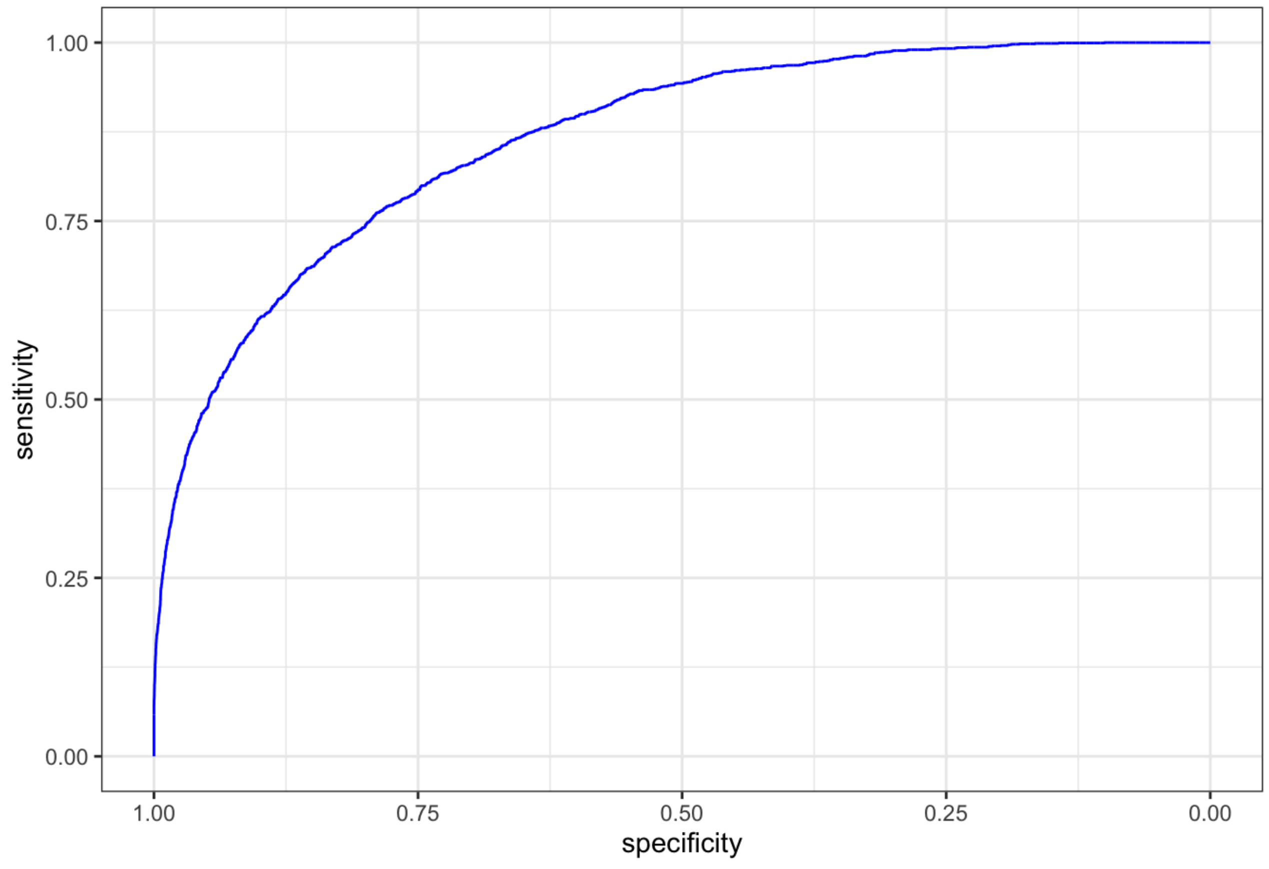
```
fit_c = glm(formula = Def_ind ~ tot_balance + num_acc_30d_past_due_12_months +
    uti_open_card + num_inq_12_month + credit_age + avg_bal_cards +
    num_acc_30d_past_due_6_months + ind_XYZ + num_card_12_month +
    num_mortgage_currently_past_due + pct_over_50_uti + num_card_inq_24_month +
    tot_amount_currently_past_due + rep_education + num_acc_30d_past_due_12_months:num_inq
_12_month +
    tot_balance:avg_bal_cards + uti_open_card:ind_XYZ + avg_bal_cards:num_card_12_month +
    uti_open_card:num_inq_12_month + tot_balance:num_acc_30d_past_due_6_months +
    tot_balance:credit_age + uti_open_card:num_card_inq_24_month +
    num_inq_12_month:avg_bal_cards + num_card_12_month:rep_education +
    num_inq_12_month:num_mortgage_currently_past_due + credit_age:num_card_12_month +
    tot_balance:num_card_12_month + I(tot_balance ^ 2) + I(avg_bal_cards^3), family = "bin
omial", data = dat)
fit_roc = roc(dat$Def_ind, fit_c$fitted.values, levels=c("0", "1"))
```

```
## Setting direction: controls < cases
```

```
fit_roc$auc
```

```
## Area under the curve: 0.8654
```

```
ggroc(fit_roc, color="blue") + theme_bw() + labs(caption = fit_roc$auc)
```



0.86535715923653

```
glm.probs =predict(fit_c, dat_test, type="response")
probs = ifelse(glm.probs > 0.5, 1, 0)
a <- table(probs)
accuracy = a[1] / (a[1]+a[2])
```

Neural Network

```
library(keras)
scaled.dat = scale(dat[, c(-21, -20, -18)])
scaled.test.dat = scale(dat_test[, c(-21, -20, -18)])
dat.y = to_categorical(dat$Def_ind, 2)
test.dat.y = to_categorical(dat_test$Def_ind, 2)
```

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 16, activation = "relu", input_shape = c(18)) %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 8, activation = "tanh") %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 2, activation = "softmax")
```

```
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## dense_2 (Dense)                     (None, 16)                      304
## ________________________________________________________________________________
## dropout_1 (Dropout)                 (None, 16)                      0
## ________________________________________________________________________________
## dense_1 (Dense)                     (None, 8)                       136
## ________________________________________________________________________________
## dropout (Dropout)                   (None, 8)                       0
## ________________________________________________________________________________
## dense (Dense)                       (None, 2)                       18
## ================================================================================
## Total params: 458
## Trainable params: 458
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)

history <- model %>% fit(
  scaled.dat, dat.y,
  epochs = 15, batch_size = 15,
  validation_split = 0.2
)
```

4. Evaluation and Prediction on Test data

```
model %>% evaluate(scaled.test.dat, test.dat.y,verbose = 1)
```

```
##      loss  accuracy
##       NaN 0.9107286
```

```
y_pred = model %>% predict_classes(scaled.test.dat)
```