Modeling and Predictive Analysis of Credit Card Approval Outcome

ANLY 512 - Final Project 2020

Group 11
Team members:

Feiyang Jiang
Yanou Yang
Xuejun Zhang
Zihao Zhou
Directed By: Prof. Purna Gamage

Introduction

Nowadays, people use credit cards for almost everything, for example, online shopping, paying for electric bills, purchasing concert tickets and so on. Credit card issuers attract applicants by providing appealing benefits such as opening bonus and cash back by using their credit cards. However, creditors also suffer from huge losses if their customers fail to pay off the debts.

Credit risk is a huge challenge to creditors. Currently, the U.S. consumer debt was $13.7 trillion in Q1 2019, which amounts to approximately $40.0 trillion in the whole year of 2019. Also in 2019, around 60% of U.S. cardholders between 18 and 34 carried large balances and continually paid late fees, and among them, 8% are in serious delinquency. Moreover, 7 million Americans were at least 90 days behind their car loan payments("*Reduce Credit Risk* | Brighterion AI | A Mastercard Company", 2020). Creditors suffer from huge losses and need to identify the risks before they occur. This serious situation motivates our team to come up with a statistical model to assist the creditors to identify potential risky credit card applicants.

In this project, our team wanted to explore the personal background information of the applicants to further investigate the relationships between these predictors and the risk level of applicants in order to conduct predictive analysis. We will apply feature engineering to the predictors, which cover applicants' background information such as annual income, number of children, education level, in order to predict the level of risk of credit card applicants. This project will not only help the bank mitigate risk of approving unqualified applicants, but also assist other financial sectors such as personal loans and mortgage to improve their premium and borrowing models.

This project has four parts. The first part is to define the response variable. According to the Basel Accord in 2007 (Thomas, 2009), users who overdue more than 90 days are considered unable to repay their debt; therefore, in order to control the risk under 3%, we choose to use 60 days overdue as the cutoff point for the users ("high risk" if >=60, "low risk" <60). The second part is to clean the predictors since the predictors include categorical variable, binary variable, and continuous variable. The third part will be applying feature engineering, such as LASSO Regression, Principal Component Analysis, Forward and Backward selection, Classification Tree, Random Forest, and Bagging to determine the statistically significant predictors. The last part will be model evaluation. We will split the dataset into testing and training data. Among all models mentioned above, we will compare the test accuracy, recall, precision scores to select the best-fitted model for the dataset.

<u>Analysis</u>

i. Data Collection and Preprocessing

For the data collection of this project, the data we used is from Kaggle website. ("Credit Card Approval Prediction", 2020) The data contains two csv files. The first file has 18 columns/variables and 438558 rows that includes the applicants' application record: ID, Gender, ownership of car, ownership of realty, count of children, income,

income type, education type, family status, housing type, birth days, employed days, whether applicants filed the mobile phone number, whether applicants filed the work phone number, whether applicants filed the phone number, whether applicants filed their emails, occupation, and count of family members. The second file has 3 columns/variables and 1048576 rows that includes applicants' credit record:ID, month balance, and risk status. Month balance means the month of the extracted data is the starting point, backwards. For example, in month balance, "0" is the current month, "-1" is the previous month, and so on. Status explains the applicants' payments overdue situation and it has the following rules: "0" means 1-29 days past due; "1" means 30-59 days past due; "2" means 60-89 days overdue; "3" means 90-119 days overdue; "4" means 120-149 days overdue; "5" means over 150 days overdue which means the bad debt; "C" means paid off that month and "X" means no loan for the month.

To preprocess the two dataset, we first used the largest absolute value in the month balance column from the second data file to find the applicant credit history in month and then created a new column named Begin_Month. Then, we joined two files on the applicants' ID. Rows with missing values are dropped. Next, the target column named risk is created by determining whether the applicant has payment overdue. If a applicant' status shows the applicant has no loan, all loans paid off, and less than 60 days overdue, he will be marked as lower risk applicant which is "0" in the risk column; if a applicant' status shows the applicant has payments overdue 60 days, he will be marked as higher risk applicant. Then, since begin_month and risk columns are created, the month balance and status columns are dropped. Birth days and employed days in days unit are transformed to ages and employed years in years unit. All the categorical values are converted to numerical values, such as code_gender column in male and female are transformed to “1” as female and “0” as male. In addition, some numerical columns’ values are distributed too widely, so bins are created to replace the original columns. For example, the income column varies from $27000 to $1575000, so there are three bins in $0 - $40k, $40k - $100k, $100k - $2000k corresponding to “1”, “2”, “3” created. Finally, it is found that the dataset is very imbalanced, because there are only 2957 lower risk applicants and 24, 404 higher risk applicants. The synthetic minority oversampling technique abbreviated as SMOTE is used to oversampled and undersampled the lower risk applicants rows and higher risk applicants rows. In the end, the dataset the team used has 31 columns/variables and 20700 rows.

ii. Methodology

SMOTE

As mentioned in the data preprocessing section, SMOTE is used to undersample the lower risk applicants and oversample the higher risk applicants. In SMOTE, undersampling the majority class is intuitive, and oversampling the minority class uses k-nearest neighbors to control the way the new examples are created. ("SMOTE function | R Documentation", 2020) Precisely, in the dataset, for each originally existing lower risk applicant class example X, new examples will be created. These new examples will be generated using the information from the k nearest neighbors of each lower risk applicant. In this project, k of 5 is used.

K-means Clustering

Clustering seeks a partition of the data into distinct groups so that the observations within each group are quite similar to each other. K-means Clustering seeks to partition the observations into a pre-specified number of clusters.

The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation is as small as possible.

Details of K-means clustering algorithm(James, Witten, Hastie and Tibshirani, n.d.):

Algorithm: K-Means

1.Randomly assign a number, from 1 to K, to each of the observations. There serve as initial cluster assignments for the observations.

2.Iterate until the cluster assignments stop changing:

2.1 For each of the K clusters, compute the cluster centroid. The Kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.

2.2 Assign each observation to the cluster whose centroid is closest(where closest is defined using Euclidean distance).

K-means clustering assumes spherical shapes of different clusters of data and works badly at non-spherical shapes of clusters. But K-means provide a good sketch to better understand the inner relationship of the dataset.

PCA

PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variable that have maximal variance, and are mutually uncorrelated. Through it, we can directly decrease the number of feature variables, thereby narrowing down the important features and saving on computations.

PCA mainly has three steps:

1. Compute the covariance matrix of the data
2. Compute the eigenvalues and eigenvectors of the covariance matrix
3. Use the eigenvalues and eigenvectors to select only the most important principle components and transform your data into uncorrelated linear combinations to reduce dimensionality.

PCA not only can do dimension reduction, but also can provide good visualization and understanding of the dataset. To this dataset, PCA primarily works as this job.

Forward Selection and Backward Selection

Forward selection is a type of stepwise regression which has 4 steps:

1. Start with no variables in the model
2. Test the addition of each variable using a chosen model fit criterion
3. Add the variable (if any) whose inclusion gives the most statistically significant improvement of the fit
4. Repeat this process until none improves the model to a statistically significant extent.

Backward selection is a type of stepwise regression which has 4 steps:

1. Involve starting with all candidate variables
2. Testing the deletion of each variable using a chosen model fit criterion
3. Deleting the variable (if any) whose loss gives the most statistically insignificant deterioration of the model fit
4. Repeating this process until no further variables can be deleted without a statistically insignificant loss of fit.

In this case, forward selection and backward selection are used to select the best subset of predictors to simplify our model and prevent overfitting. C_p, BIC and adjusted R^2 are used as the criteria of these two procedures.

LASSO

Lasso is a regression analysis method that improves the prediction accuracy and interpretability of regression models by altering the model fitting process to select only a subset of the provided covariates.

Lasso uses the l1 penalty to force the sum of the absolute value of the regression coefficients to be less than a fixed value, which shrinks certain coefficients to zero, so we can choose a simpler model that does not include those coefficients. Before using lasso, we scale the variables.

Logistic Regression

Logistic regression is a kind of generalized linear model (GLM) that uses a logistic function as its link function to model a variable with multinomial distribution. Here the predictors selected by LASSO are used to build the model.

Generalized Additive Model

Generalized additive model (GAM) is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions. Also, the predictors selected by LASSO are used to build the model. Compared with the logistic regression model above, all the predictors other than the binary predictors FLAG_WORK_PHONE and FLAG_PHONE are replaced by smoothing splines.

Random Forest

Random forest is one of the tree based models which can eliminate the basic tree models' problem in high variance, especially when the dataset's ratio of n (number of observations) by p (number of variables/dimension) is large. Because the dataset has many categorical values, the random forest classification model is chosen. The random forest classification model performs following steps below (James, Witten, Hastie and Tibshirani, n.d.):

Algorithm: Random Forest in Classification
1. Use bootstrap method to take B repeated samples from the training dataset 2. Create B decision trees using the B bootstrapped samples from the last step, numbers of variables randomly sampled as candidates at each split and numbers of sampled trees need to be determined in this step. Notice here random forest is not using all of the dataset. 3. For the test dataset, record the class predicted by each of the B decision trees and take the majority vote: the overall prediction which is the most commonly occuring majority class among the B prediction will be the final random forest model.

Bagging is a special case of random forest when the number of variables randomly sampled at each split (m) in trees equals the number of variables (p) in the dataset. Normally, in random forest classification model, half of number of variables (p/2) and square root of number of variables ($\sqrt{p}$) are chosen as the number of variables randomly sampled at each split to create trees.

Results

i. EDA

Exploratory data analysis is performed to get the general pattern of some variables. Figure 1 shows the education type of the applicants distributed in our dataset. Education types “1”, “2”, “3”, “4”, “5” stand for “Incomplete Secondary”, “Secondary/secondary special”, “Incomplete higher”, “Higher education” and “Academic degree”. Figure 1 explains that most of the applicants have secondary/secondary special and higher education. Only a few applicants’ education level is in incomplete secondary and academic degrees.

Figure 2 shows the applicants’ age distribution, most of applicants are between 29-50 years, and fewer applicants are under 29 years.

Figure 3 shows the applicants’ occupation distribution, most of applicants’ work as laborers, managers, drivers, sales staff etc. The occupation types distributions look generally balanced.

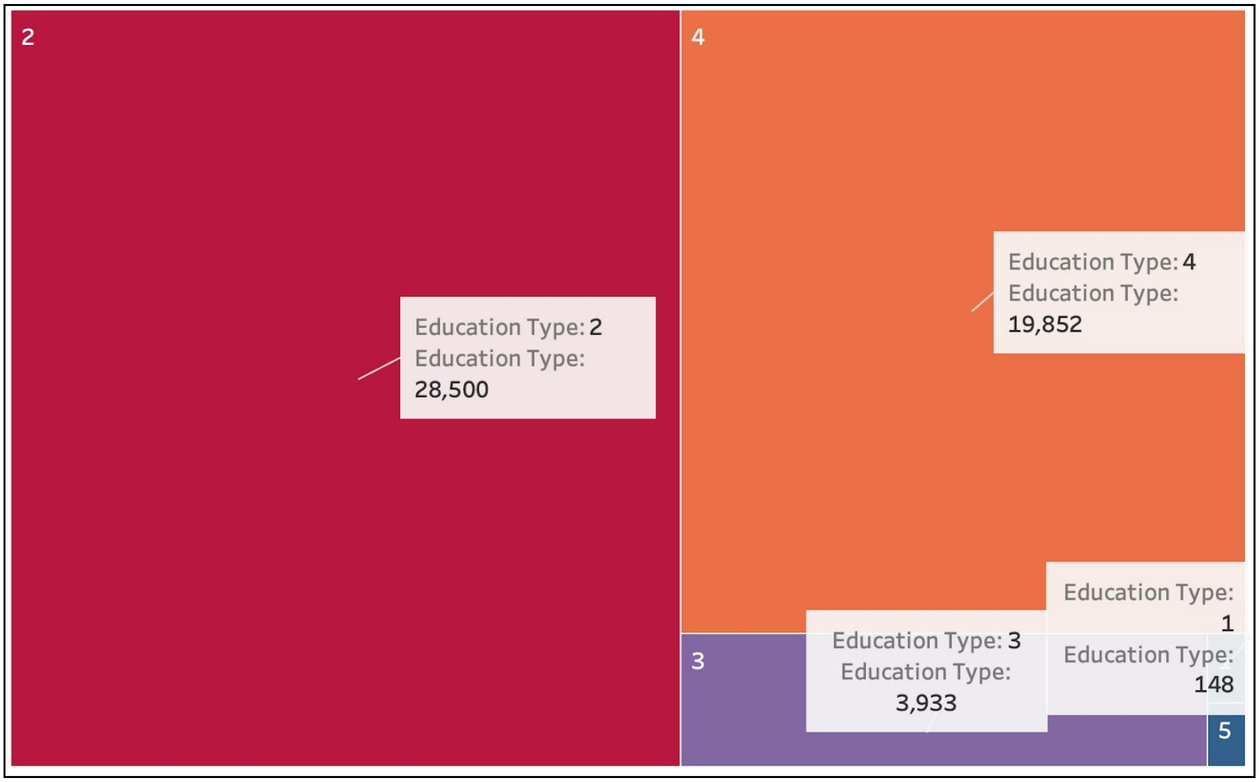


Figure 1: Applicants' Education Types Distribution

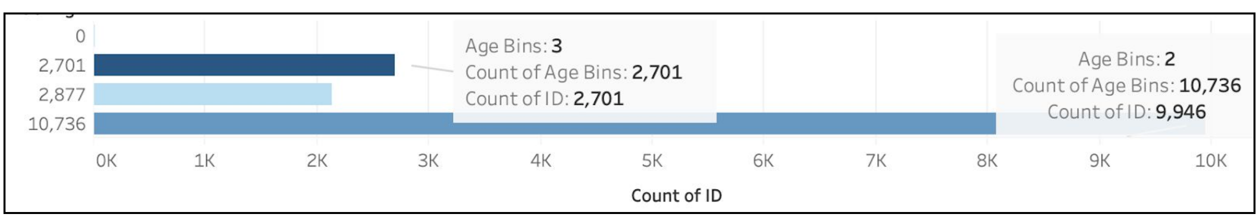


Figure 2: Applicants' Ages Distribution

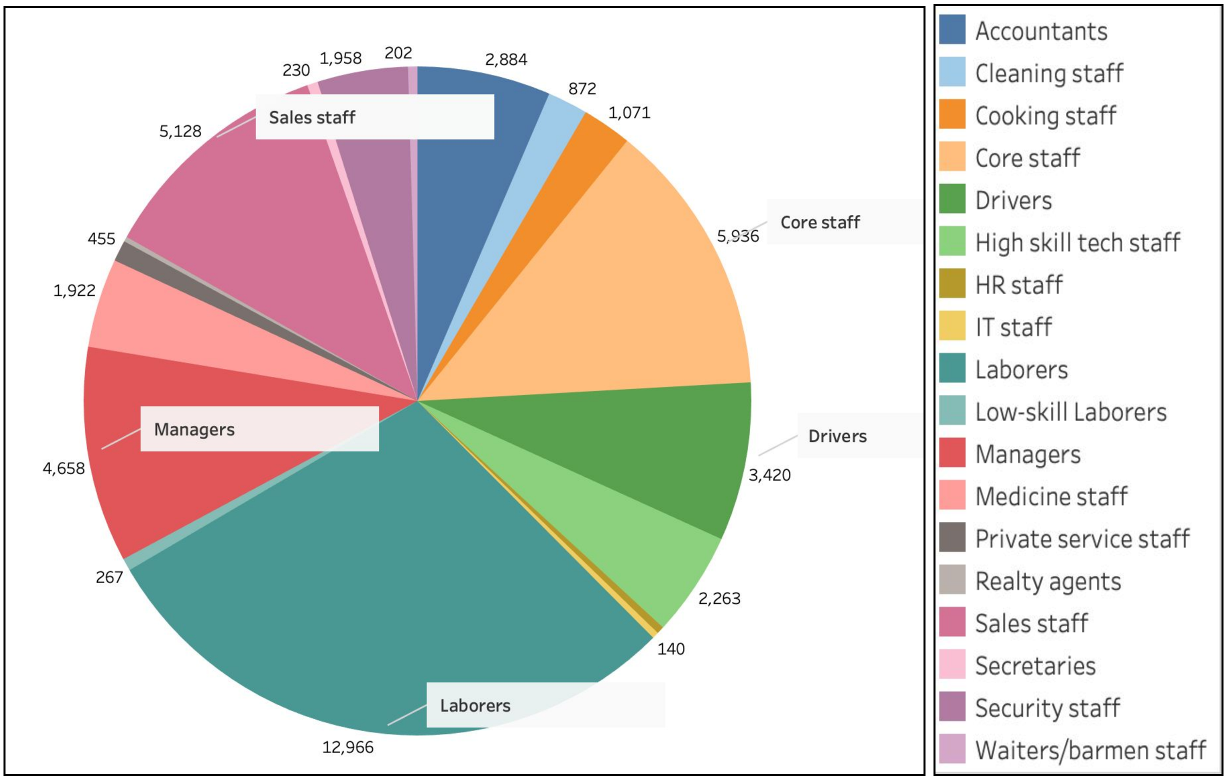


Figure 3: Applicants' Occupations Distribution

ii. Unsupervised Learning

K-means Clustering

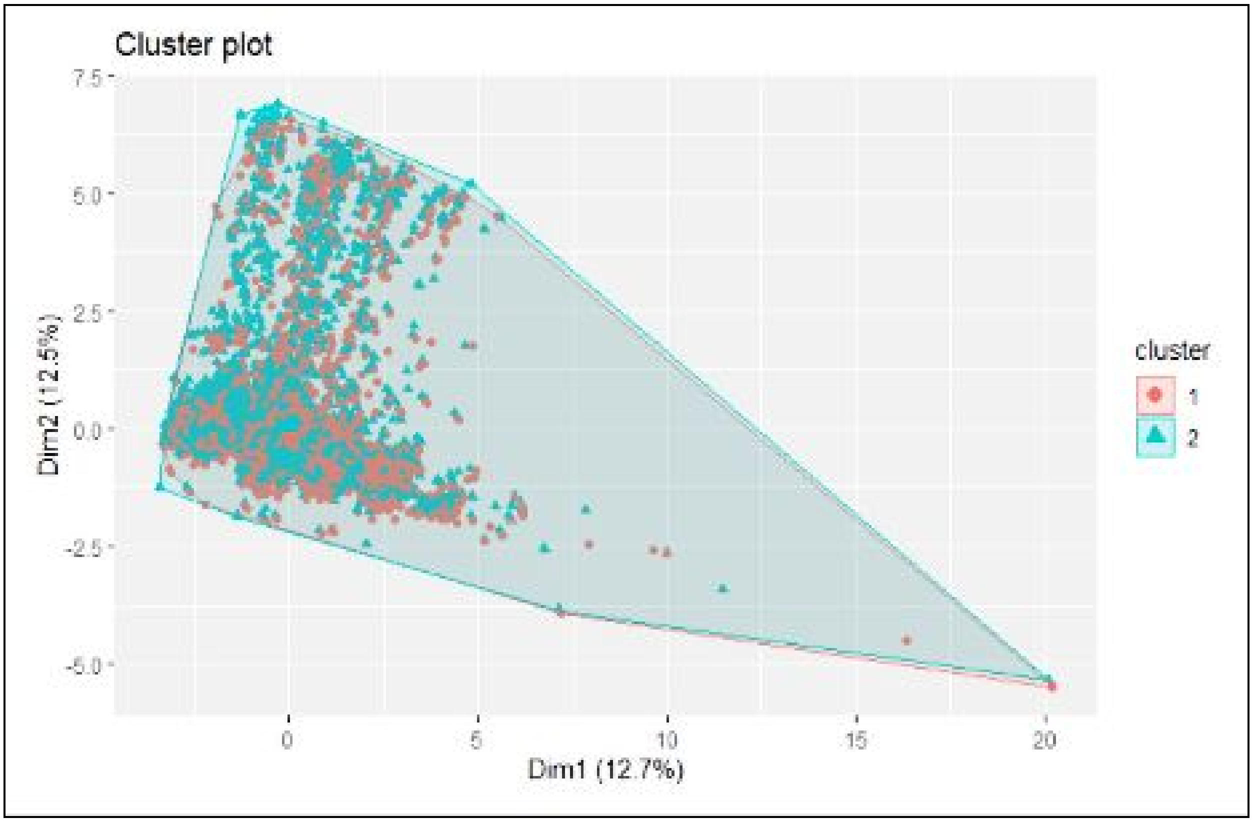


Figure 4: K-means Clustering in 2D

K-means Clustering results are plotted in the above graph, red point is cluster 1 and blue point is cluster 2. Points within cluster 1 are the most similar to each other, which means this kind of applicant has the same features, so as cluster 2.

We applied K-means Clustering with K = 2. From the plot above, it is observed that two clusters are overlapped with each other and hard to be separated by clustering. It means applicants even in different clusters still have many same features, which prevents K-means to segment the applicant from non-risky to risky. Besides, the two clusters are not-circle shaped either.

In sum, after applying the K-means Clustering method, the result shows that we need supervised learning to determine if a customer is risky.

PCA

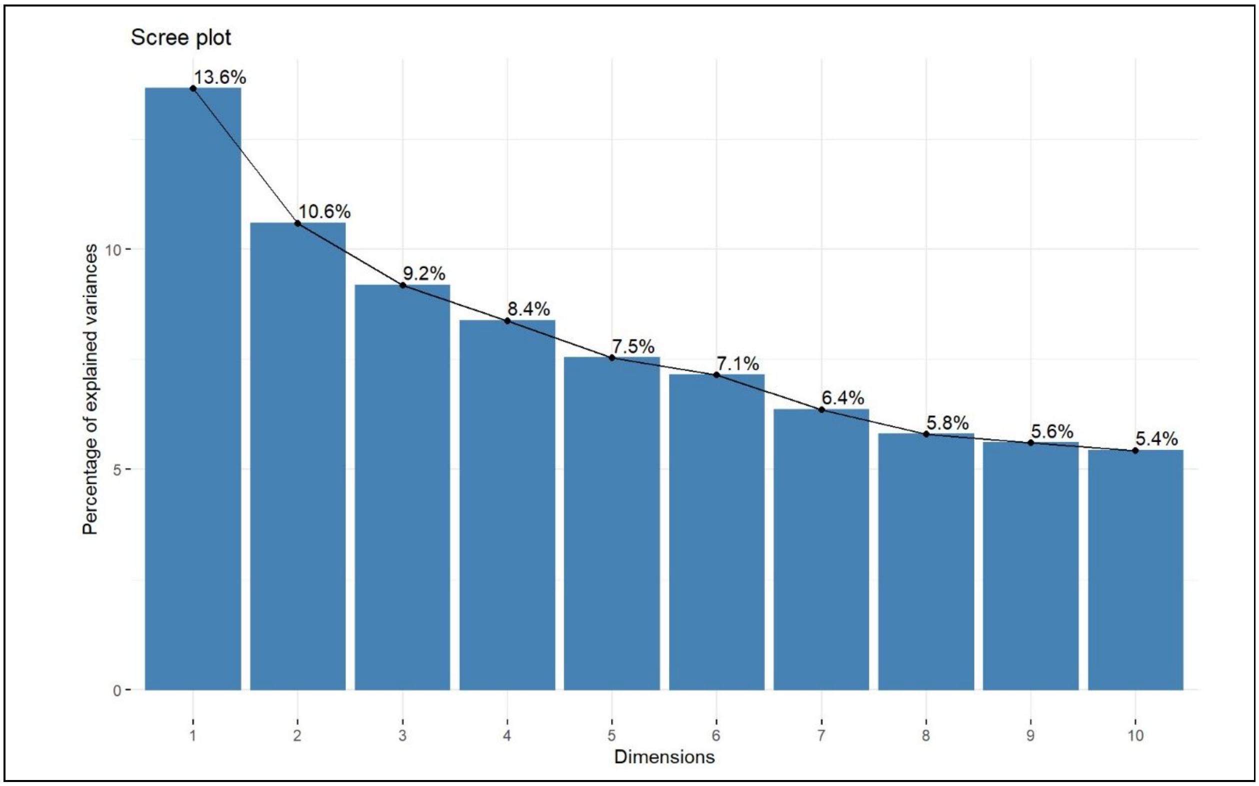


Figure 5: Percentage of Explained Variables vs. Dimensions

The above graph is the histogram of the percentage of explained variables and dimensions. From the graph, the first dimension only explains 13.6% variance. The histogram shows that even the first two principles can only explain 20% variance of the predictors, which would tell the overall movement in the original dataset. In other words, if we use principal components to determine the risk, then it would include at least 5 principal components, we are unable to reduce the original dimensions.

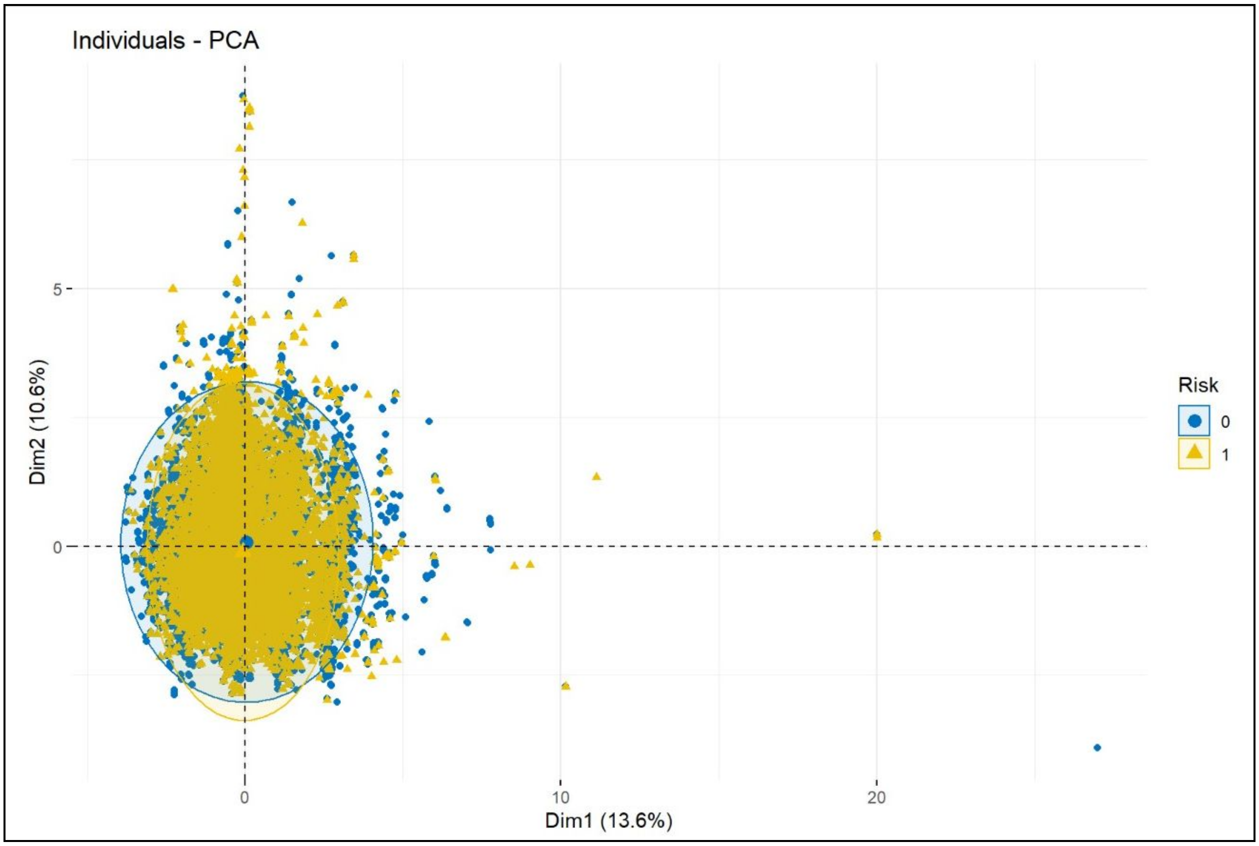


Figure 6: Individuals embedding on 2D

The above graph shows individuals embedding into 2D after PCA. From the plot, yellow points and blue points are closely clustered and overlapped.

By embedding the first two principal components into 2D plot, risk 0 and risk 1 points are overlapped as well, indicating whether being risky or not is determined by multiple principal components.

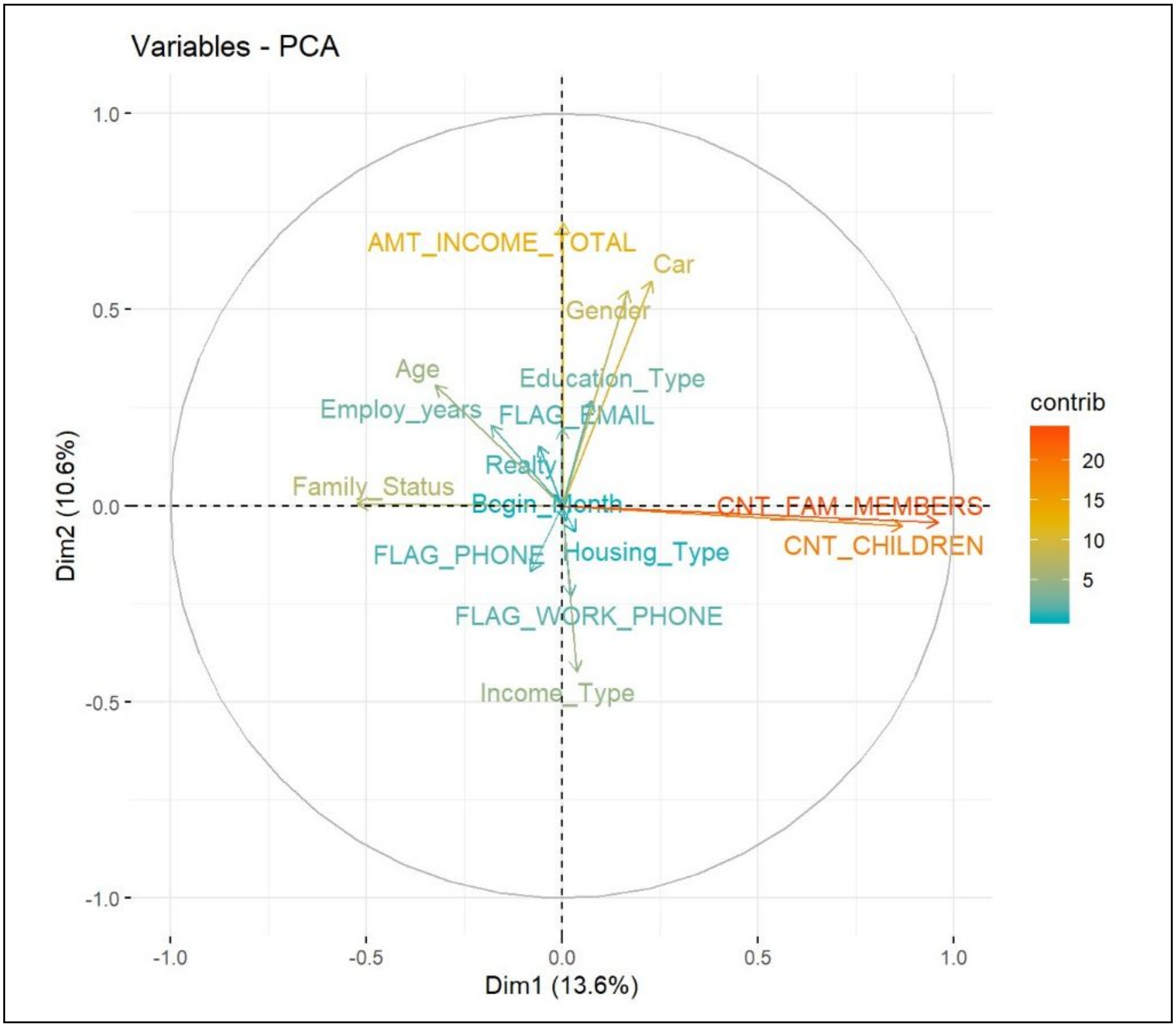


Figure 7: Variables embedding on 2D

This graph shows which variable contributes the most to explain the variance. It's obvious to find most of the variables only contribute less than 10% to the total variance.

PCA results show that variables like the number of children, the number of family members, and AMT account balance explain the most variance, which can be paid more attention to.

iii. Feature Selection

Forward Selection and Backward Selection

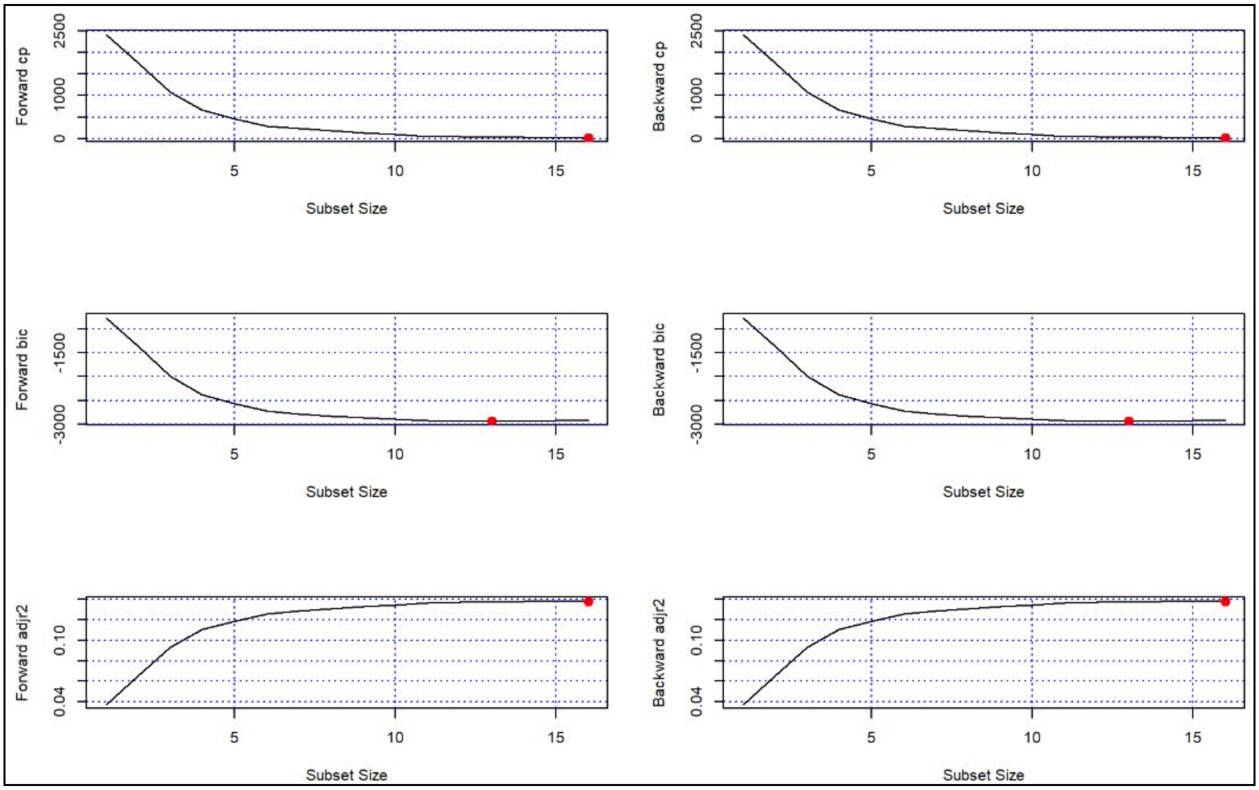


Figure 8: The forward and backward selection

The C_p and adjusted R^2 criteria suggest that we should choose a subset that includes all 16 predictors, while the BIC criteria suggests that we should choose a subset with size 14. All the criteria suggest a large subset to reach optimum, which is consistent with the finding of unsupervised learning. However, the plot shows that when the subset size is 6, the criteria is already good.

LASSO Regression

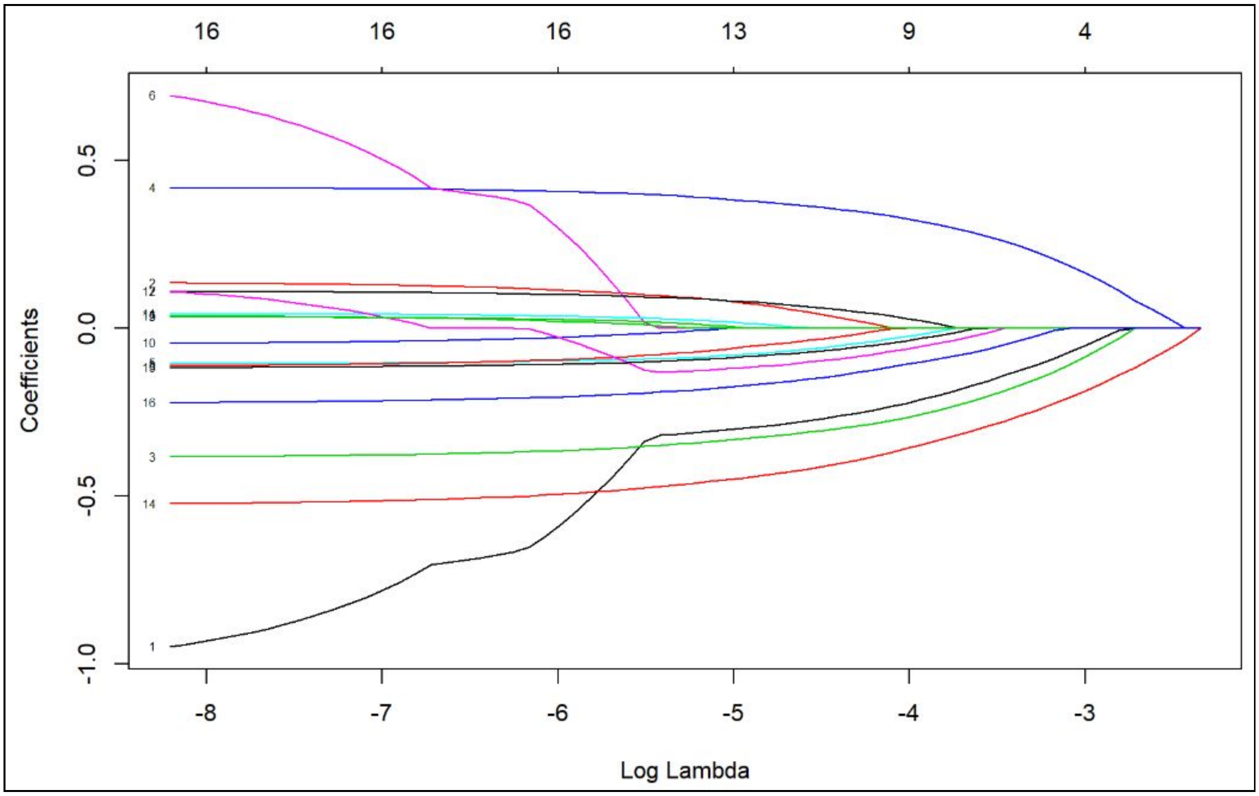


Figure 9: The coefficient selected by LASSO

At $log(lambda) = -3.5$, the LASSO selects 6 variables “Count of Children”, “Filling in Work Phone Number”, “Filling in Phone Number”, “Ages”, “Education Type” and “Family Status” as important features to determine whether the credit cards applicant has high risk, the selection is the same with forward and backward selection.

iv. Supervised Learning

Logistic Regression

```
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.69080    0.11138  24.159   <2e-16 ***
CNT_CHILDREN     -0.45058    0.02602 -17.317   <2e-16 ***
FLAG_WORK_PHONE  -0.99548    0.04571 -21.776   <2e-16 ***
FLAG_PHONE        0.98741    0.03993  24.727   <2e-16 ***
Family_Status    -0.19087    0.02142  -8.911   <2e-16 ***
Age              -0.04979    0.00191 -26.064   <2e-16 ***
Education_Type   -0.22425    0.01977 -11.340   <2e-16 ***
```

Figure 10: Coefficients for logistic regression

The p-values are all less than the significance level 0.05, suggesting that the risk is significantly correlated with all the predictors selected.

The coefficient shows that:

The applicants that are currently married have higher risks of arrear.

The applicants with more children have lower risks of arrear.

The applicants that provide work phone numbers have lower risks of arrear.

The applicants that provide phone numbers have higher risks of arrear.

The older applicants have lower risks of arrear.

The applicants with higher education level have lower risks of arrear.

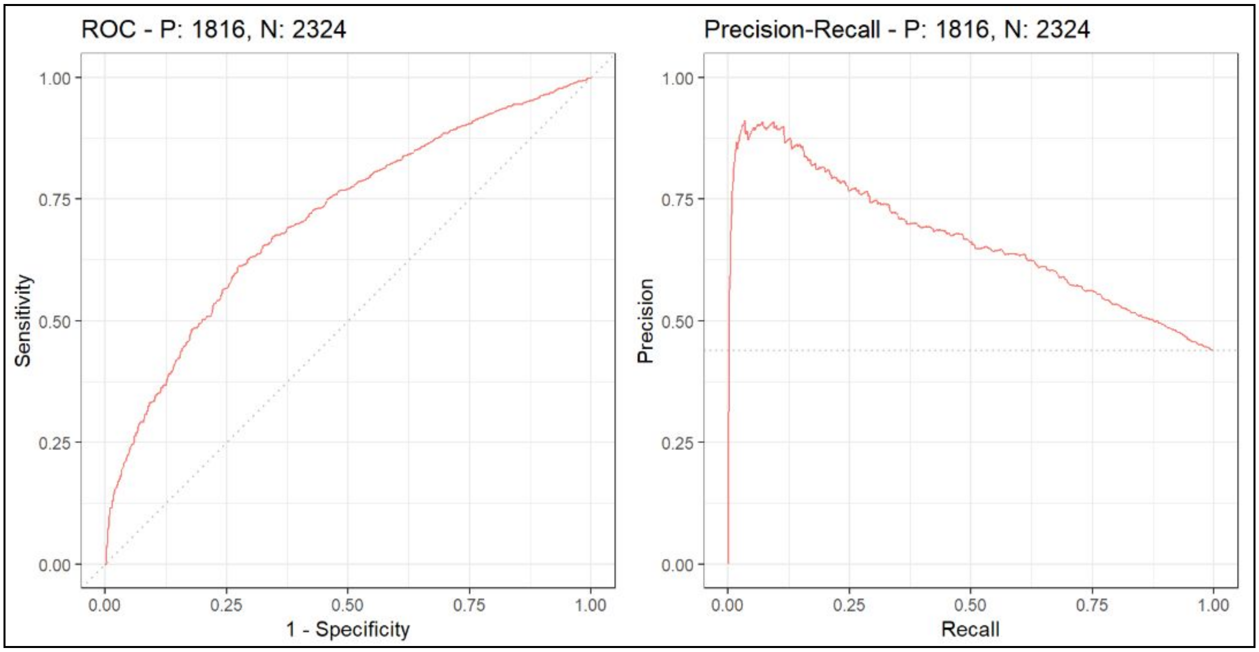


Figure 11: The ROC and precision-recall curve for Logistic Regression

The ROC and precision-recall curve show a tradeoff between true positive rate and false positive rate and between precision and recall. The precision score here is the proportion of the applicants that truly have risks among the predicted applicants with risks, and recall is the proportion of applicants with risks that are correctly identified by the model among the applicants that truly have risks. In this case, our goal is to identify the applicants with risks, therefore, we are more concerned about the recall in the precision-recall tradeoff.

Under a 0.4 threshold, the recall is 69.93%, and the precision is 57.89%.

Generalized Additive Model with Smoothing Spline

Anova for Parametric Effects

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
s(CNT_CHILDREN)	1	213.3	213.29	211.727	< 2.2e-16	***
FLAG_WORK_PHONE	1	186.0	185.97	184.604	< 2.2e-16	***
FLAG_PHONE	1	536.9	536.88	532.947	< 2.2e-16	***
s(Family_Status)	1	47.9	47.92	47.572	5.494e-12	***
s(Age)	1	560.7	560.67	556.568	< 2.2e-16	***
s(Education_Type)	1	128.2	128.24	127.303	< 2.2e-16	***
Residuals	16540	16662.0	1.01			

Figure 11: The ANOVA table of Generalized Additive Model with Smoothing Spline

The p-values of ANOVA are all less than the significance level 0.05, suggesting that all the predictors selected have significant influence on the risk.

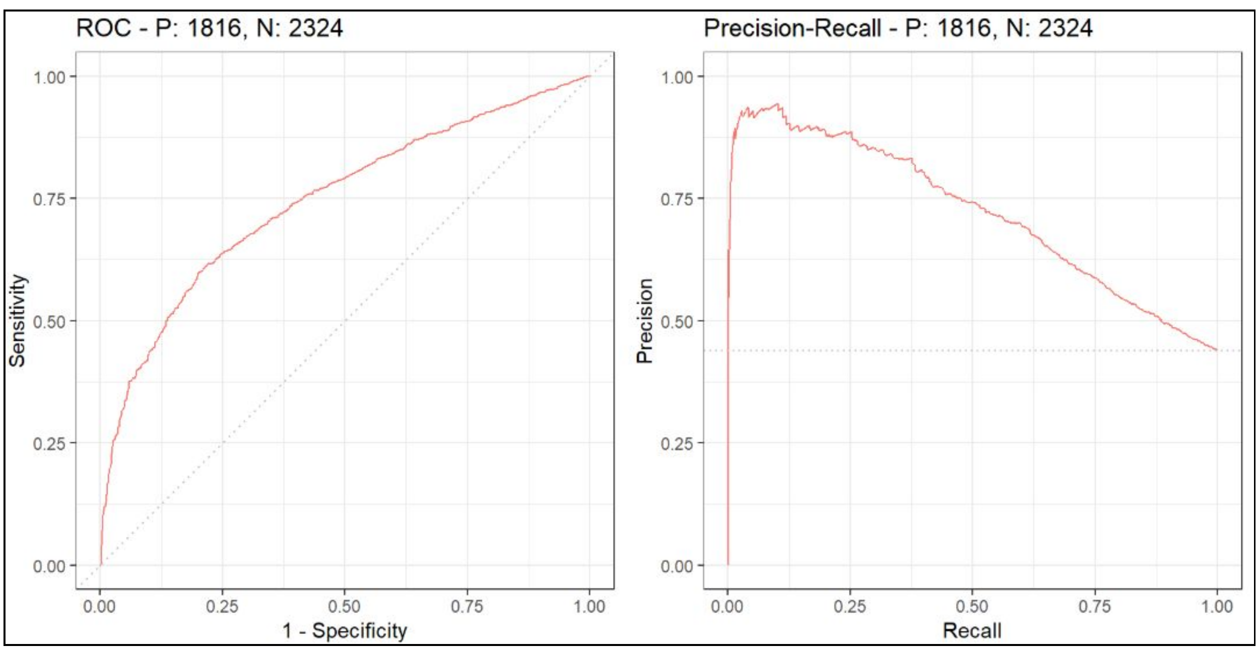


Figure 12:The ROC and precision-recall curve for GAM model with Smoothing Splines

The above plots are the ROC and precision-recall curve, which shows that recall and precision of the generalized additive model is a little higher. For a 0.4 threshold, the recall is 71.20%, and the precision is 60.51%, which is a little better. The logistic regression failed to give a higher recall probably because there are still some binary categorical predictors in this model, although they can be transformed into dummy variables, the performance of the model will be influenced.

Random Forest

Three random forests models are performed to the dataset. 17 variables are used as predictors. The response variable is the risk level of the applicants. Predictors include count of children, count of family members, ages, genders and so on. As shown in Table 1, the bagging model is performed when the number of variables randomly sampled at each split (Mtry) equalsl to the number of predictors 17. The train recall, precision and accuracy are 83.51%, 70.22% and 81.40% respectively. As the Mtry is changed to 17/2, the random forest model has train recall, precision and accuracy of 85.69%, 70.39% and 82.34% respectively. As the Mtry is changed to sqrt(17) , 4 variables randomly sampled at each split in sample trees and the random forest model has train recall, precision and accuracy of 93.43%, 68.40% and 84.49% respectively. In Table2, The test recall, precision and accuracy are 85.67%, 71.75% and 83.09% respectively. As the Mtry is changed to 17/2, the random forest model has test recall, precision and accuracy of 85.61%, 72.25% and 83.00% respectively. As the Mtry is changed to sqrt(17) , 4 variables randomly sampled at each split in sample trees and the random forest model has test recall, precision and accuracy of 94.48%, 70.04% and 84.93% respectively. Test data recall, precision and accuracy are all higher than train data recall, precision and accuracy, so there's no evidence of overfitting. Model with the highest recall is selected which is highlighted in Table 1 and Table 2. Table 3 shows the optimal model the team chose in r, and 40,000 of trees sampled, to ensure that every input row gets predicted at least a few times. Recall in 94.28% means that about 94% of applicants with actual risk can be captured by model. Precision in 70.93% means that about 70% of applicants with true high risk situations under the higher risks predicted class. Accuracy in 84.93% means about 85% of applicants are predicted correctly in the model.

Table 1: Different Trees Models Train Results by Setting the Same Ntree = 500

Model	Mtry	Train Recall	Train Precision	Train Accuracy
Bagging	17	83.51%	70.22%	81.40%
Random Forest	17/2	85.69%	70.39%	82.34%
Random Forest	**sqrt(17)**	**93.43%**	**68.40%**	**84.49%**

Table 2: Different Trees Models Test Results by Setting the Same Ntree = 500

Model	Mtry	Test Recall	Test Precision	Test Accuracy
Bagging	17	85.67%	71.75%	83.09%

Random Forest	17/2	85.68%	72.25%	83.00%
Random Forest	**sqrt(17)**	**94.48%**	**70.04%**	**84.93%**

Table 3: The Final Optimal Random Forest Model Settings and Result

Model	Ntree	Mtry	Test Recall	Test Precision	Test Accuracy
Random Forest	40,000	sqrt(17)	94.28%	70.93%	84.95%

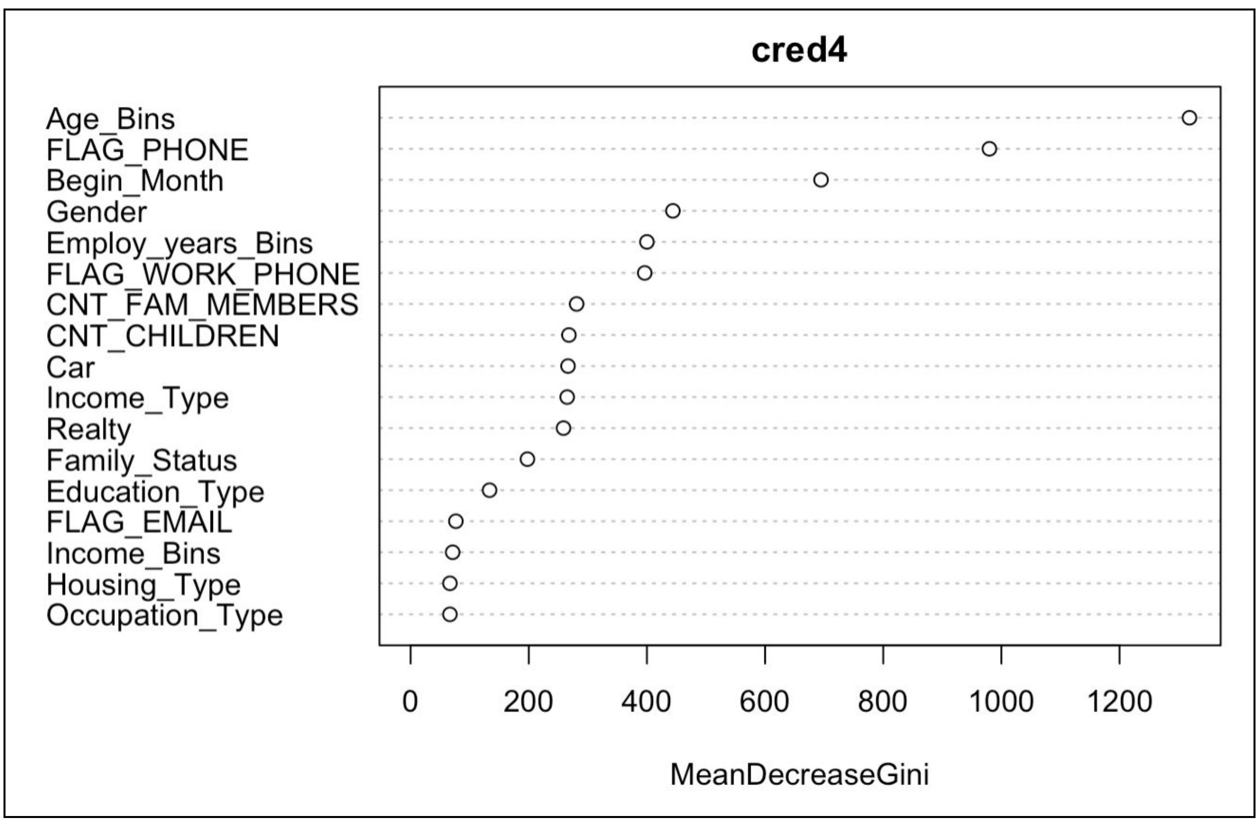


Figure 13: Feature Importance Plot of the Final Optimal Random Forest Model

From the feature importance plot of the final optimal random forest model, applicants' ages, we noticed whether filing the phone numbers, credit history in months, genders and employed years are the top important variables in classifying whether the applicant has high risk or not.

Conclusions

In conclusion, the statistically significant features we selected are *education level*, *whether filled in work phone number*, *family status*, *number of children*, and *employment history*, and *age*. We have four major findings from the above modeling and analysis. First of all, we would like to advise creditors to be cautious with applicants with higher education levels. This is a surprising finding because we would consider people with higher education levels to earn more; therefore they are more likely to repay their debts. However, according to the article, "*1 million people default on their student loans each year*" by CNBC, "Within four years after leaving school, nearly a quarter of the borrowers had defaulted", and this article also found that those people are in financial distress (Nova, 2020). Although people with higher education may have the ability to repay their debts, the student loans they have when they pursue higher education make them more vulnerable to financial burdens.

Secondly, we would like to advise creditors to be cautious with applicants with short employment history. Similar to the reasons stated in the previous finding that applicants with short employment history may suffer from problems with low income, short credit history, unpaid student loans (Nova, 2020). Therefore, they have a higher probability of defaulting compared to people who have accumulated wealth for several years.

Thirdly, applicants who provide work phone numbers are found to be low-risk. This finding is intuitive because if the applicants provide work phone numbers, then it indicates that they have a stable job and consistent income every month; therefore, they are more likely to fulfill their debt responsibility.

Lastly, we would like to advise creditors to be cautious with applicants with no or few children. This is a surprising finding as well because we would think families with more children would have more financial burdens and less likely to be able to repay their debts. According to the article, "*Families With Young Kids Are More Likely to Live in Poverty*", it states that families with few children tend to be young parents(Ryan, 2020). Due to the lack of workplace support and income to young parents and lack of experience in raising young children, they are more likely to be in poverty(Ryan, 2020). Therefore, they are less likely to repay debts on time.

In all, we hope this project would help creditors and financial sectors such as mortgage and personal loans to mitigate risk of approving unqualified candidates by analyzing the background information they provide in the application process.

Discussion

We faced several challenges in completing this project. First of all, the original dataset is highly unbalanced that only 10% of applicants are low-risk and nearly 90% of applicants are high-risk. When we first fit the logistic model to the data, the accuracy is as high as 87%, but both recall and precision scores are very low. Therefore, we had to resample the data to achieve better recall and precision scores. The second difficulty we faced was to choose the number of predictors. Although the forward and the backward selection chose 14-17 predictors, we decided to build a simpler model with 6 predictors because the Cp scores were not improved much after adding predictors. Interpreting the model is also challenging because some of the findings are counterintuitive. We researched through a good amount of related works and academic papers to understand why our results align with the reality.

The next steps for this project are to reassess the risk level of applicants by incorporating more factors such as demographic background, whether the applicants have car loans, how many credit cards do applicants already have etc... We will also be cautious in overfitting the model because adding more predictors are not necessarily better in predicting the outcomes. We will continue to research more about creditors' current risk models because having risky customers can also be profitable for creditors since they will pay late fees and penalties, so we are curious how creditors distinguish the applicants who default and applicants who are risky but profitable to creditors.

References

James, G., Witten, D., Hastie, T. and Tibshirani, R., n.d. *An Introduction To Statistical Learning*.

Reduce Credit Risk | Brighterion AI | A Mastercard Company. (2020). Retrieved 29 April 2020, from https://brighterion.com/reduce-credit-risk-with-ai/?gclid=CjwKCAjw4pT1BRBUEiwAm5QuRyL1HWOW16l8gDFwNFxyXfZI3evYmB-Chxo3fml7buNudCbRt0SWJB0CO88QAvD_BwE

Nova, A. (2020). *More than 1 million people default on their student loans each year*. Retrieved 29 April 2020, from https://www.cnbc.com/2018/08/13/twenty-two-percent-of-student-loan-borrowers-fall-into-default.html

Ryan, L. (2020). *Families With Young Kids Are More Likely to Live in Poverty*. Retrieved 29 April 2020, from https://www.thecut.com/2016/12/families-with-young-children-have-lower-incomes-report.html

Credit Card Approval Prediction. (2020). Retrieved 29 April 2020, from https://www.kaggle.com/rikdifos/credit-card-approval-prediction#application_record.csv

SMOTE function | R Documentation. (2020). Retrieved 29 April 2020, from https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE

512 Proj DC

Xuejun Zhang

4/18/2020

```
##Read originial files
appred <- read.csv("/Users/YukiZ./Desktop/512/credit-card-approval-prediction/application_record.csv")
crered <- read.csv("/Users/YukiZ./Desktop/512/credit-card-approval-prediction/credit_record.csv")
```

```
##Preview
library(dplyr)
```

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

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

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

```
glimpse(appred)
```

```
## Observations: 438,557
## Variables: 18
## $ ID                  <int> 5008804, 5008805, 5008806, 5008808, 5008809, 5008…
## $ CODE_GENDER         <fct> M, M, M, F, F, F, F, F, F, F, M, M, M, M, M, M, M…
## $ FLAG_OWN_CAR        <fct> Y, Y, Y, N, N, N, N, N, N, N, Y, Y, Y, Y, Y, Y, Y…
## $ FLAG_OWN_REALTY     <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y…
## $ CNT_CHILDREN        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ AMT_INCOME_TOTAL    <dbl> 427500, 427500, 112500, 270000, 270000, 270000, 2…
## $ NAME_INCOME_TYPE    <fct> Working, Working, Working, Commercial associate, …
## $ NAME_EDUCATION_TYPE <fct> Higher education, Higher education, Secondary / s…
## $ NAME_FAMILY_STATUS  <fct> Civil marriage, Civil marriage, Married, Single /…
## $ NAME_HOUSING_TYPE   <fct> Rented apartment, Rented apartment, House / apart…
## $ DAYS_BIRTH          <int> -12005, -12005, -21474, -19110, -19110, -19110, -…
## $ DAYS_EMPLOYED       <int> -4542, -4542, -1134, -3051, -3051, -3051, -3051, …
## $ FLAG_MOBIL          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ FLAG_WORK_PHONE     <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ FLAG_PHONE          <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ FLAG_EMAIL          <int> 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0…
## $ OCCUPATION_TYPE     <fct> , , Security staff, Sales staff, Sales staff, Sal…
## $ CNT_FAM_MEMBERS     <dbl> 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2…
```

```
#appred
```

```
glimpse(crered)
```

```
## Observations: 1,048,575
## Variables: 3
## $ ID             <int> 5001711, 5001711, 5001711, 5001711, 5001712, 5001712, …
## $ MONTHS_BALANCE <int> 0, -1, -2, -3, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, …
## $ STATUS         <fct> X, 0, 0, 0, C, C, C, C, C, C, C, C, C, 0, 0, 0, 0, 0, …
```

```
summary(crered)
```

```
##        ID          MONTHS_BALANCE       STATUS
##  Min.   :5001711   Min.   :-60.00   C      :442031
##  1st Qu.:5023644   1st Qu.:-29.00   0      :383120
##  Median :5062104   Median :-17.00   X      :209230
##  Mean   :5068286   Mean   :-19.14   1      : 11090
##  3rd Qu.:5113856   3rd Qu.: -7.00   5      :  1693
##  Max.   :5150487   Max.   :  0.00   2      :   868
##                                     (Other):   543
```

```
#crered
```

```
##find all users' account open month
Begin_Month <- summarise_at(group_by(crered,ID),vars(MONTHS_BALANCE),funs(min(.,na.rm=TRUE)))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
names(Begin_Month)[2] <- "Begin_Month"
Begin_Month
```

```
## # A tibble: 45,985 x 2
##         ID Begin_Month
##      <int>       <int>
##  1 5001711          -3
##  2 5001712         -18
##  3 5001713         -21
##  4 5001714         -14
##  5 5001715         -59
##  6 5001717         -21
##  7 5001718         -38
##  8 5001719         -42
##  9 5001720         -35
## 10 5001723         -30
## # … with 45,975 more rows
```

```
##Merge begin_month to application record data
appred1 <- merge(appred, Begin_Month, by="ID")
#appred1
```

```
#label risk (y)
##0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overd
ue 5: Overdue or bad debts, write-offs for more than 150 days ,C trans to 7: paid off that month X trans to 8: No
 loan for the month
##0 is nonrisk; 1 is under risk
crered$Risk = NULL
crered$STATUS <- as.numeric(crered$STATUS)
crered$Risk <- as.numeric(crered$STATUS > 1 & crered$STATUS < 6)
#crered
```

```
appred2 <- merge(appred1, crered, by="ID")
appred2$MONTHS_BALANCE <- NULL
appred2$STATUS <- NULL
#appred2
```

```
#Drop rows with no OCCUPATION_TYPE values in appred3
occu_none <- appred2[appred2$OCCUPATION_TYPE == '',]
appred3 <- subset(appred2, appred2$OCCUPATION_TYPE != '')
appred3 <- na.omit(appred3)
#appred3
```

```
#Add Gender Index
appred4 = mutate(appred3, Gender = as.numeric(appred3$CODE_GENDER))
#Add Have Car or Not Index
appred5 = mutate(appred4, Car = as.numeric(appred4$FLAG_OWN_CAR))
#Add Property or Not Index
appred6 = mutate(appred5, Realty = as.numeric(appred5$FLAG_OWN_REALTY))
#Add Income Type Index
appred7 = mutate(appred6, Income_Type = as.numeric(appred6$NAME_INCOME_TYPE))
#Add Family Status Index
appred8 = mutate(appred7, Family_Status = as.numeric(appred7$NAME_FAMILY_STATUS))
#Add Housing Type Index
appred9 = mutate(appred8, Housing_Type = as.numeric(appred8$NAME_HOUSING_TYPE))
#Add Occupation Type Index
appred10 = mutate(appred9, Occupation_Type = as.numeric(appred9$NAME_HOUSING_TYPE))
```

```
glimpse(appred10)
```

```
## Observations: 537,667
## Variables: 27
## $ ID                  <int> 5008806, 5008806, 5008806, 5008806, 5008806, 5008…
## $ CODE_GENDER         <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M…
## $ FLAG_OWN_CAR        <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y…
## $ FLAG_OWN_REALTY     <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y…
## $ CNT_CHILDREN        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ AMT_INCOME_TOTAL    <dbl> 112500, 112500, 112500, 112500, 112500, 112500, 1…
## $ NAME_INCOME_TYPE    <fct> Working, Working, Working, Working, Working, Work…
## $ NAME_EDUCATION_TYPE <fct> Secondary / secondary special, Secondary / second…
## $ NAME_FAMILY_STATUS  <fct> Married, Married, Married, Married, Married, Marr…
## $ NAME_HOUSING_TYPE   <fct> House / apartment, House / apartment, House / apa…
## $ DAYS_BIRTH          <int> -21474, -21474, -21474, -21474, -21474, -21474, -…
## $ DAYS_EMPLOYED       <int> -1134, -1134, -1134, -1134, -1134, -1134, -1134, …
## $ FLAG_MOBIL          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ FLAG_WORK_PHONE     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ FLAG_PHONE          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ FLAG_EMAIL          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ OCCUPATION_TYPE     <fct> Security staff, Security staff, Security staff, S…
## $ CNT_FAM_MEMBERS     <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Begin_Month         <int> -29, -29, -29, -29, -29, -29, -29, -29, -29, -29,…
## $ Risk                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ Gender              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Car                 <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Realty              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Income_Type         <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5…
## $ Family_Status       <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Housing_Type        <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Occupation_Type     <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
```

```
summary(appred10)
```

```
##        ID          CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  CNT_CHILDREN
##  Min.   :5008806   F:333832    N:306207     N:192196        Min.   : 0.0000
##  1st Qu.:5044925   M:203835    Y:231460     Y:345471        1st Qu.: 0.0000
##  Median :5079091                                            Median : 0.0000
##  Mean   :5079231                                            Mean   : 0.5067
##  3rd Qu.:5115755                                            3rd Qu.: 1.0000
##  Max.   :5150487                                            Max.   :19.0000
##
##  AMT_INCOME_TOTAL              NAME_INCOME_TYPE
##  Min.   :  27000   Commercial associate:151412
##  1st Qu.: 135000   Pensioner           :   332
##  Median : 180000   State servant       : 52733
##  Mean   : 197117   Student             :   322
##  3rd Qu.: 229500   Working             :332868
##  Max.   :1575000
##
##                      NAME_EDUCATION_TYPE            NAME_FAMILY_STATUS
##  Academic degree              :   434   Civil marriage      : 44083
##  Higher education             :153770   Married             :384003
##  Incomplete higher            : 20590   Separated           : 31394
##  Lower secondary              :  4556   Single / not married: 65944
##  Secondary / secondary special:358317   Widow               : 12243
##
##
##           NAME_HOUSING_TYPE    DAYS_BIRTH     DAYS_EMPLOYED      FLAG_MOBIL
##  Co-op apartment    :  3396   Min.   :-24611   Min.   :-15713   Min.   :1
##  House / apartment  :474177   1st Qu.:-17594   1st Qu.: -3661   1st Qu.:1
##  Municipal apartment: 18023   Median :-14785   Median : -2147   Median :1
##  Office apartment   :  4159   Mean   :-15011   Mean   : -2762   Mean   :1
##  Rented apartment   :  8561   3rd Qu.:-12239   3rd Qu.: -1050   3rd Qu.:1
##  With parents       : 29351   Max.   : -7489   Max.   :   -17   Max.   :1
##
##  FLAG_WORK_PHONE    FLAG_PHONE       FLAG_EMAIL
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.2816   Mean   :0.2989   Mean   :0.1007
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##              OCCUPATION_TYPE   CNT_FAM_MEMBERS   Begin_Month
##  Laborers             :131572   Min.   : 1.000   Min.   :-60.00
##  Core staff           : 77112   1st Qu.: 2.000   1st Qu.:-47.00
##  Sales staff          : 70362   Median : 2.000   Median :-36.00
##  Managers             : 67738   Mean   : 2.303   Mean   :-34.66
##  Drivers              : 47678   3rd Qu.: 3.000   3rd Qu.:-23.00
##  High skill tech staff: 31768   Max.   :20.000   Max.   :  0.00
##  (Other)              :111437
##       Risk              Gender           Car            Realty
##  Min.   :0.00000   Min.   :1.000   Min.   :1.00   Min.   :1.000
##  1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:1.00   1st Qu.:1.000
##  Median :0.00000   Median :1.000   Median :1.00   Median :2.000
##  Mean   :0.01357   Mean   :1.379   Mean   :1.43   Mean   :1.643
##  3rd Qu.:0.00000   3rd Qu.:2.000   3rd Qu.:2.00   3rd Qu.:2.000
##  Max.   :1.00000   Max.   :2.000   Max.   :2.00   Max.   :2.000
##
##   Income_Type    Family_Status   Housing_Type   Occupation_Type
##  Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:2.00   1st Qu.:2.000   1st Qu.:2.000
##  Median :5.000   Median :2.00   Median :2.000   Median :2.000
##  Mean   :3.675   Mean   :2.29   Mean   :2.309   Mean   :2.309
##  3rd Qu.:5.000   3rd Qu.:2.00   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :5.000   Max.   :5.00   Max.   :6.000   Max.   :6.000
##
```

```
#Create Bins for Income Total
appred10$Income_Bins <- cut(appred10$AMT_INCOME_TOTAL, breaks=c(0,40000,100000,2000000), labels=c("1","2","3"))
```

```
#Convert Days Birth to Age
appred11 = mutate(appred10, Age = round((-1*(appred10$DAYS_BIRTH)/365), digits = 0))
appred11$DAYS_BIRTH <- NULL
#appred11
#Create Bins for Age
appred11$Age_Bins <- cut(appred11$Age, breaks=c(0,29,50,80), labels=c("1","2","3"))
```

```
#Convert DAYS_EMPLOYED to Years
appred12 = mutate(appred11, Employ_years = round((-1*(appred11$DAYS_EMPLOYED)/365), digits = 0))
appred12$DAYS_EMPLOYED <- NULL
#Create Bins for Employ Years
appred12$Employ_years_Bins <- cut(appred12$Employ_years, breaks=c(0,5,12,50), labels=c("1","2","3"))
```

```
#Add Education Type Index
appred13 = mutate(appred12, Education_Type = as.numeric(appred12$NAME_EDUCATION_TYPE))
appred13$Begin_Month = -1*appred13$Begin_Month #How many months ago
glimpse(appred13)
```

```
## Observations: 537,667
## Variables: 31
## $ ID                  <int> 5008806, 5008806, 5008806, 5008806, 5008806, 5008…
## $ CODE_GENDER         <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M…
## $ FLAG_OWN_CAR        <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y…
## $ FLAG_OWN_REALTY     <fct> Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y, Y…
## $ CNT_CHILDREN        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ AMT_INCOME_TOTAL    <dbl> 112500, 112500, 112500, 112500, 112500, 112500, 1…
## $ NAME_INCOME_TYPE    <fct> Working, Working, Working, Working, Working, Work…
## $ NAME_EDUCATION_TYPE <fct> Secondary / secondary special, Secondary / second…
## $ NAME_FAMILY_STATUS  <fct> Married, Married, Married, Married, Married, Marr…
## $ NAME_HOUSING_TYPE   <fct> House / apartment, House / apartment, House / apa…
## $ FLAG_MOBIL          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ FLAG_WORK_PHONE     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ FLAG_PHONE          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ FLAG_EMAIL          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ OCCUPATION_TYPE     <fct> Security staff, Security staff, Security staff, S…
## $ CNT_FAM_MEMBERS     <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Begin_Month         <dbl> 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 2…
## $ Risk                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ Gender              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Car                 <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Realty              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Income_Type         <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5…
## $ Family_Status       <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Housing_Type        <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Occupation_Type     <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
## $ Income_Bins         <fct> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3…
## $ Age                 <dbl> 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 59, 5…
## $ Age_Bins            <fct> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3…
## $ Employ_years        <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3…
## $ Employ_years_Bins   <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ Education_Type      <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5…
```

```
appred13 = appred13[complete.cases(appred13), ]
appred13 = unique(appred13)
#appred13
write.csv(appred13,"/Users/YukiZ./Desktop/512/credit-card-approval-prediction/CleanV2.csv", row.names = FALSE)
```

final_tryout

Xuejun Zhang

4/25/2020

```r
library(dplyr)
```

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

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

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

```r
da <- read.csv("/Users/YukiZ./Desktop/512/credit-card-approval-prediction/CleanV3.csv")
```

```r
glimpse(da)
```

```
## Observations: 20,699
## Variables: 31
## $ ID                  <dbl> 5024068, 5069020, 5106043, 5023948, 5104958, 5022…
## $ CODE_GENDER         <fct> M, F, F, M, M, F, M, F, F, M, F, F, F, F, M, F…
## $ FLAG_OWN_CAR        <fct> Y, N, N, Y, Y, Y, N, N, N, N, N, N, Y, Y, N, N, N…
## $ FLAG_OWN_REALTY     <fct> Y, Y, N, Y, Y, Y, N, Y, Y, Y, N, Y, Y, Y, N, Y, N…
## $ CNT_CHILDREN        <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 0, 0, 1…
## $ AMT_INCOME_TOTAL    <dbl> 270000, 202500, 157500, 148500, 180000, 202500, 1…
## $ NAME_INCOME_TYPE    <fct> Working, Working, State servant, Working, Working…
## $ NAME_EDUCATION_TYPE <fct> Secondary / secondary special, Secondary / second…
## $ NAME_FAMILY_STATUS  <fct> Married, Married, Civil marriage, Married, Marrie…
## $ NAME_HOUSING_TYPE   <fct> House / apartment, House / apartment, Municipal a…
## $ FLAG_MOBIL          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ FLAG_WORK_PHONE     <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ FLAG_PHONE          <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0…
## $ FLAG_EMAIL          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0…
## $ OCCUPATION_TYPE     <fct> Laborers, Laborers, Core staff, Laborers, Drivers…
## $ CNT_FAM_MEMBERS     <dbl> 2, 2, 2, 3, 2, 1, 2, 1, 2, 2, 3, 1, 4, 3, 2, 1, 2…
## $ Begin_Month         <dbl> 6, 0, 17, 14, 21, 32, 16, 38, 49, 46, 28, 15, 10,…
## $ Risk                <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ Gender              <dbl> 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1…
## $ Car                 <dbl> 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1…
## $ Realty              <dbl> 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1…
## $ Income_Type         <dbl> 5, 5, 3, 5, 5, 5, 5, 5, 3, 5, 1, 1, 5, 1, 1, 3, 3…
## $ Family_Status       <dbl> 2, 2, 1, 2, 2, 5, 2, 4, 2, 2, 2, 4, 2, 2, 2, 4, 4…
## $ Housing_Type        <dbl> 2, 2, 3, 2, 2, 2, 2, 2, 6, 3, 6, 2, 2, 2, 2, 2, 2…
## $ Occupation_Type     <dbl> 2, 2, 3, 2, 2, 2, 2, 2, 6, 3, 6, 2, 2, 2, 2, 2, 2…
## $ Income_Bins         <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 3, 3…
## $ Age                 <dbl> 39, 56, 50, 41, 59, 33, 50, 51, 30, 49, 49, 27, 3…
## $ Age_Bins            <dbl> 2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2…
## $ Employ_years        <dbl> 11, 10, 8, 14, 7, 4, 20, 5, 8, 15, 5, 7, 1, 7, 6,…
## $ Employ_years_Bins   <dbl> 2, 2, 2, 3, 2, 1, 3, 1, 2, 3, 1, 2, 1, 2, 2, 2, 2…
## $ Education_Type      <int> 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
```

```r
summary(da$Risk==0)
```

```
##    Mode   FALSE    TRUE
## logical    8871   11828
```

```r
da$CODE_GENDER <- NULL
da$FLAG_MOBIL <- NULL
da$FLAG_OWN_CAR <- NULL
da$FLAG_OWN_REALTY <- NULL
da$AMT_INCOME_TOTAL <-NULL
da$NAME_INCOME_TYPE <-NULL
da$NAME_FAMILY_STATUS <- NULL
da$NAME_HOUSING_TYPE <- NULL
da$OCCUPATION_TYPE <- NULL
da$Age <- NULL
da$Employ_years <-NULL
da$ID <-NULL
```

```r
da$Education_Type <-NULL
transform_education_type = function(df, x){
  if (df[x] == "Academic degree"){
    return(5)
    }else if(df[x] == "Higher education"){
    return(4)
    }else if(df[x] == "Incomplete higher"){
    return(3)
    }else if(df[x] == "Secondary / secondary special"){
    return(2)
    }else{
    return(1)
    }
}
da$Education_Type = apply(da, 1, transform_education_type, x="NAME_EDUCATION_TYPE")
```

```r
da$NAME_EDUCATION_TYPE <- NULL
summary(da)
```

```
##   CNT_CHILDREN     FLAG_WORK_PHONE    FLAG_PHONE       FLAG_EMAIL
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.0000   Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   : 0.4362   Mean   :0.2316   Mean   :0.3579   Mean   :0.09461
##  3rd Qu.: 1.0000   3rd Qu.:0.2638   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :19.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  CNT_FAM_MEMBERS   Begin_Month         Risk            Gender
##  Min.   : 1.000   Min.   : 0.00   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 2.000   1st Qu.:15.00   1st Qu.:0.0000   1st Qu.:1.000
##  Median : 2.000   Median :26.00   Median :0.0000   Median :1.000
##  Mean   : 2.251   Mean   :26.90   Mean   :0.4286   Mean   :1.354
##  3rd Qu.: 2.975   3rd Qu.:37.57   3rd Qu.:1.0000   3rd Qu.:2.000
##  Max.   :20.000   Max.   :60.00   Max.   :1.0000   Max.   :2.000
##       Car            Realty       Income_Type    Family_Status
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.049   1st Qu.:2.000
##  Median :1.000   Median :2.000   Median :5.000   Median :2.000
##  Mean   :1.407   Mean   :1.664   Mean   :3.727   Mean   :2.274
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :5.000   Max.   :5.000
##   Housing_Type   Occupation_Type  Income_Bins      Age_Bins
##  Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:3.00   1st Qu.:1.611
##  Median :2.000   Median :2.000   Median :3.00   Median :2.000
##  Mean   :2.294   Mean   :2.294   Mean   :2.91   Mean   :1.912
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:3.00   3rd Qu.:2.000
##  Max.   :6.000   Max.   :6.000   Max.   :3.00   Max.   :3.000
##  Employ_years_Bins Education_Type
##  Min.   :1.000     Min.   :1.000
##  1st Qu.:1.000     1st Qu.:2.000
##  Median :1.262     Median :2.000
##  Mean   :1.625     Mean   :2.545
##  3rd Qu.:2.000     3rd Qu.:3.000
##  Max.   :3.000     Max.   :5.000
```

```r
glimpse(da)
```

```
## Observations: 20,699
## Variables: 18
## $ CNT_CHILDREN      <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1, 0, 0, 1, …
## $ FLAG_WORK_PHONE   <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ FLAG_PHONE        <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, …
## $ FLAG_EMAIL        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, …
## $ CNT_FAM_MEMBERS   <dbl> 2, 2, 2, 3, 2, 1, 2, 1, 2, 2, 3, 1, 4, 3, 2, 1, 2, …
## $ Begin_Month       <dbl> 6, 0, 17, 14, 21, 32, 16, 38, 49, 46, 28, 15, 10, 8…
## $ Risk              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ Gender            <dbl> 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, …
## $ Car               <dbl> 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, …
## $ Realty            <dbl> 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 1, …
## $ Income_Type       <dbl> 5, 5, 3, 5, 5, 5, 5, 5, 3, 5, 1, 1, 5, 1, 1, 3, 3, …
## $ Family_Status     <dbl> 2, 2, 1, 2, 2, 5, 2, 4, 2, 2, 2, 4, 2, 2, 2, 4, 4, …
## $ Housing_Type      <dbl> 2, 2, 3, 2, 2, 2, 2, 2, 6, 3, 6, 2, 2, 2, 2, 2, 2, …
## $ Occupation_Type   <dbl> 2, 2, 3, 2, 2, 2, 2, 2, 6, 3, 6, 2, 2, 2, 2, 2, 2, …
## $ Income_Bins       <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 3, 3, …
## $ Age_Bins          <dbl> 2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2, …
## $ Employ_years_Bins <dbl> 2, 2, 2, 3, 2, 1, 3, 1, 2, 3, 1, 2, 1, 2, 2, 2, 2, …
## $ Education_Type    <dbl> 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
```

```r
summary(da)
```

```
##   CNT_CHILDREN     FLAG_WORK_PHONE    FLAG_PHONE       FLAG_EMAIL
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.0000   Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   : 0.4362   Mean   :0.2316   Mean   :0.3579   Mean   :0.09461
##  3rd Qu.: 1.0000   3rd Qu.:0.2638   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :19.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  CNT_FAM_MEMBERS   Begin_Month         Risk            Gender
##  Min.   : 1.000   Min.   : 0.00   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 2.000   1st Qu.:15.00   1st Qu.:0.0000   1st Qu.:1.000
##  Median : 2.000   Median :26.00   Median :0.0000   Median :1.000
##  Mean   : 2.251   Mean   :26.90   Mean   :0.4286   Mean   :1.354
##  3rd Qu.: 2.975   3rd Qu.:37.57   3rd Qu.:1.0000   3rd Qu.:2.000
##  Max.   :20.000   Max.   :60.00   Max.   :1.0000   Max.   :2.000
##       Car            Realty       Income_Type    Family_Status
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.049   1st Qu.:2.000
##  Median :1.000   Median :2.000   Median :5.000   Median :2.000
##  Mean   :1.407   Mean   :1.664   Mean   :3.727   Mean   :2.274
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :5.000   Max.   :5.000
##   Housing_Type   Occupation_Type  Income_Bins      Age_Bins
##  Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:3.00   1st Qu.:1.611
##  Median :2.000   Median :2.000   Median :3.00   Median :2.000
##  Mean   :2.294   Mean   :2.294   Mean   :2.91   Mean   :1.912
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:3.00   3rd Qu.:2.000
##  Max.   :6.000   Max.   :6.000   Max.   :3.00   Max.   :3.000
##  Employ_years_Bins Education_Type
##  Min.   :1.000     Min.   :1.000
##  1st Qu.:1.000     1st Qu.:2.000
##  Median :1.262     Median :2.000
##  Mean   :1.625     Mean   :2.545
##  3rd Qu.:2.000     3rd Qu.:3.000
##  Max.   :3.000     Max.   :5.000
```

```r
#write.csv(da,"/Users/YukiZ./Desktop/512/credit-card-approval-prediction/CleanV3.csv", row.names = FALSE)
```

```r
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
set.seed(100)
da$Risk = as.factor(da$Risk)
train = sample(nrow(da), 0.8*nrow(da), replace = FALSE)
ori.train = da[train, -7]
ori.test = da[-train, -7]
risk.train = da[train, "Risk"]
risk.test = da[-train,"Risk" ]
```

```r
set.seed(100)
cred1 <- randomForest(ori.train, y = risk.train, xtest = ori.test , ytest = risk.test, mtry = 17, ntree = 500)
cred2 <- randomForest(ori.train, y = risk.train , xtest = ori.test , ytest = risk.test, mtry = 17/ 2, ntree = 500
)
cred3 <- randomForest(ori.train, y = risk.train, xtest = ori.test, ytest = risk.test, mtry = sqrt(17), ntree = 50
0)
```

```r
set.seed(100)
cred1$test$confusion
```

```
##      0    1 class.error
## 0 2106  218  0.09380379
## 1  513 1303  0.28248899
```

```r
cred2$test$confusion
```

```
##      0    1 class.error
## 0 2137  187  0.08046472
## 1  516 1300  0.28414097
```

```r
cred3$test$confusion
```

```
##      0    1 class.error
## 0 2250   74  0.03184165
## 1  550 1266  0.30286344
```

```r
cred1
```

```
##
## Call:
##  randomForest(x = ori.train, y = risk.train, xtest = ori.test,      ytest = risk.test, ntree = 500, mtry = 17)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 18.59%
## Confusion matrix:
##      0    1 class.error
## 0 8526  978    0.102904
## 1 2101 4954    0.297803
##                 Test set error rate: 17.66%
## Confusion matrix:
##      0    1 class.error
## 0 2106  218  0.09380379
## 1  513 1303  0.28248899
```

```r
cred2
```

```
##
## Call:
##  randomForest(x = ori.train, y = risk.train, xtest = ori.test,      ytest = risk.test, ntree = 500, mtry = 17/
2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 17.66%
## Confusion matrix:
##      0    1 class.error
## 0 8669  835  0.08785774
## 1 2089 4966  0.29610206
##                 Test set error rate: 16.98%
## Confusion matrix:
##      0    1 class.error
## 0 2137  187  0.08046472
## 1  516 1300  0.28414097
```

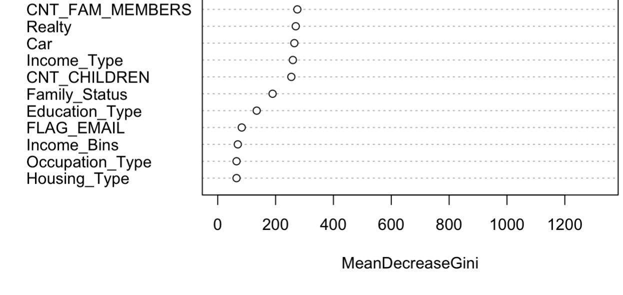
```r
cred3
```

```
##
## Call:
##  randomForest(x = ori.train, y = risk.train, xtest = ori.test,      ytest = risk.test, ntree = 500, mtry = sqr
t(17))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 15.51%
## Confusion matrix:
##      0    1 class.error
## 0 9165  339  0.03566919
## 1 2229 4826  0.31594614
##                 Test set error rate: 15.07%
## Confusion matrix:
##      0    1 class.error
## 0 2250   74  0.03184165
## 1  550 1266  0.30286344
```

```r
set.seed(100)
cred4 <- randomForest(ori.train, y = risk.train, xtest = ori.test, ytest = risk.test, mtry = sqrt(17), ntree=4000
0)
cred4
```

```
##
## Call:
##  randomForest(x = ori.train, y = risk.train, xtest = ori.test,      ytest = risk.test, ntree = 40000, mtry = s
qrt(17))
##                Type of random forest: classification
##                      Number of trees: 40000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 15.45%
## Confusion matrix:
##      0    1 class.error
## 0 9174  330  0.03472222
## 1 2228 4827  0.31580439
##                 Test set error rate: 15.05%
## Confusion matrix:
##      0    1 class.error
## 0 2248   76  0.03270224
## 1  547 1269  0.30121145
```

```r
varImpPlot(cred4)
```




```r
da$Occupation_Type <- NULL
da$Housing_Type <- NULL
da$Income_Bins <- NULL
da$FLAG_EMAIL <- NULL
da$Education_Type <- NULL
da$Family_Status <- NULL
da$Realty <-NULL
da$Income_Type <-NULL
da$Car <- NULL
da$CNT_CHILDREN <- NULL
da$FLAG_WORK_PHONE <- NULL
```

```r
library(dplyr)
glimpse(da)
```

```
## Observations: 20,699
## Variables: 7
## $ FLAG_PHONE        <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, …
## $ CNT_FAM_MEMBERS   <dbl> 2, 2, 2, 3, 2, 1, 2, 1, 2, 2, 3, 1, 4, 3, 2, 1, 2, …
## $ Begin_Month       <dbl> 6, 0, 17, 14, 21, 32, 16, 38, 49, 46, 28, 15, 10, 8…
## $ Risk              <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
## $ Gender            <dbl> 2, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, …
## $ Age_Bins          <dbl> 2, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2, …
## $ Employ_years_Bins <dbl> 2, 2, 2, 3, 2, 1, 3, 1, 2, 3, 1, 2, 1, 2, 2, 2, 2, …
```

```r
library(randomForest)
library(reprtree)
```

```
## Loading required package: tree
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
## Loading required package: plotrix
```

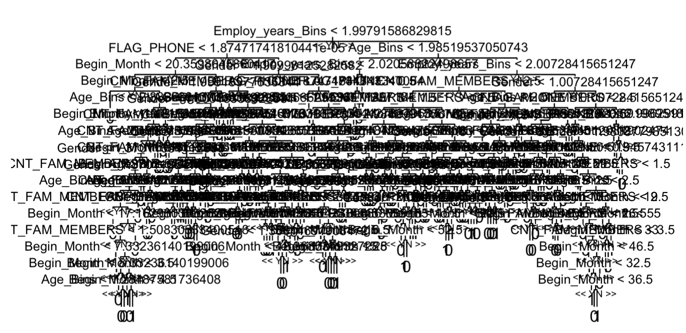
```
## Registered S3 method overwritten by 'reprtree':
##   method    from
##   text.tree tree
```

```r
model <- randomForest(Risk ~ ., data=da[-train,], importance=TRUE, ntree=4000, mtry = 2, do.trace=500)
```

```
## ntree      OOB      1      2
##   500:  14.90%  0.60% 33.20%
##  1000:  14.93%  0.56% 33.31%
##  1500:  14.93%  0.56% 33.31%
##  2000:  14.95%  0.60% 33.31%
##  2500:  14.95%  0.60% 33.31%
##  3000:  14.95%  0.60% 33.31%
##  3500:  14.93%  0.56% 33.31%
##  4000:  14.93%  0.56% 33.31%
```

```r
reprtree:::plot.getTree(model, 1, labelVar=TRUE)
```

```
## Warning in text.default(xy$x[leaves], xy$y[leaves] - 0.5 * charht, labels =
## stat, : "labelVar" is not a graphical parameter
```




```r
#reprtree:::plot.getTree(randomForest(da[-train, -7], da[-train,"Risk"], ntree = 10), k, labelVar = TRUE)
```

```r
library(rpart)                       # Popular decision tree algorithm
library(rattle)                  # Fancy tree plot
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```r
library(rpart.plot)              # Enhanced tree plots
library(RColorBrewer)                # Color selection for fancy tree plot
library(party)                   # Alternative decision tree algorithm
```

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

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```r
library(partykit)                # Convert rpart object to BinaryTree
```

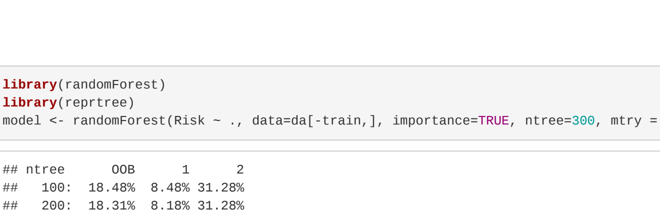
```
## Loading required package: libcoin
```

```
##
## Attaching package: 'partykit'
```

```
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner, node_surv,
##     node_terminal, varimp
```

```r
library(caret)
```

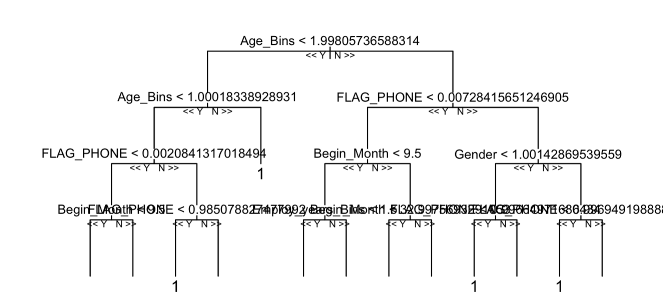
```r
library(randomForest)
library(reprtree)
model <- randomForest(Risk ~ ., data=da[-train,], importance=TRUE, ntree=1, mtry = sqrt(17))
reprtree:::plot.getTree(model, depth=5,  main= "Random Forest Tree Demo")
```




```r
library(randomForest)
library(reprtree)
model <- randomForest(Risk ~ ., data=da[-train,], importance=TRUE, ntree=300, mtry = sqrt(17), do.trace=100)
```

```
## ntree      OOB      1      2
##   100:  18.48%  8.48% 31.28%
##   200:  18.31%  8.18% 31.28%
##   300:  18.41%  8.30% 31.33%
```

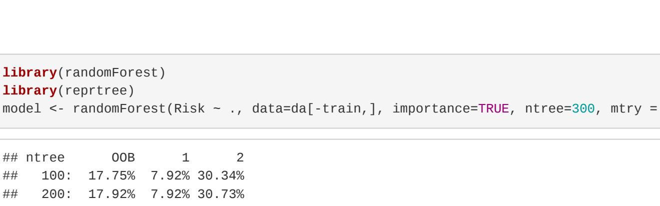
```r
reprtree:::plot.getTree(model, depth=5,  main= "Random Forest Tree Demo")
```




```r
library(randomForest)
library(reprtree)
model <- randomForest(Risk ~ ., data=da[-train,], importance=TRUE, ntree=300, mtry = sqrt(17), do.trace=100)
```

```
## ntree      OOB      1      2
##   100:  17.75%  7.92% 30.34%
##   200:  17.92%  7.92% 30.73%
##   300:  18.00%  7.92% 30.89%
```

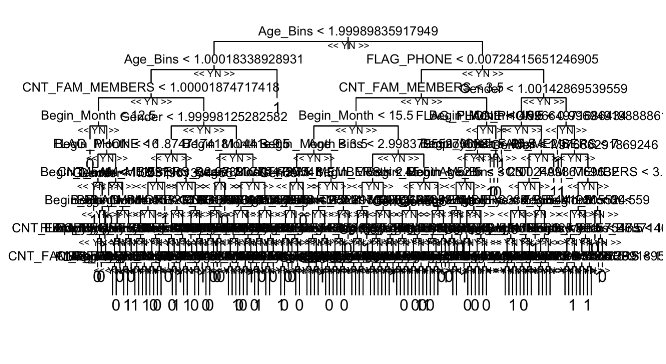
```r
reprtree:::plot.getTree(model, depth=4,  main= "Random Forest Tree Demo")
```




```r
library(randomForest)
library(reprtree)
model <- randomForest(Risk ~ ., data=da[-train,], importance=TRUE, ntree=4000, mtry = sqrt(17), do.trace=500)
```

```
## ntree      OOB      1      2
##   500:  18.09%  8.00% 31.00%
##  1000:  18.14%  8.35% 30.67%
##  1500:  18.26%  8.48% 30.78%
##  2000:  18.26%  8.39% 30.89%
##  2500:  18.21%  8.30% 30.89%
##  3000:  18.21%  8.39% 30.78%
##  3500:  18.14%  8.22% 30.84%
##  4000:  18.19%  8.26% 30.89%
```

```r
reprtree:::plot.getTree(model, depth=10,  main= "Random Forest Tree Demo")
```



project

YanouY

2020/4/23

```
data = read.csv("CleanV2.csv")
```

```
#install.packages('kernlab')
#install.packages('e1071')
#install.packages('factoextra')
```

```
library(tidyverse)  # data manipulation and visualization
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## √ ggplot2 3.2.1     √ purrr   0.3.3
## √ tibble  3.0.1     √ dplyr   0.8.4
## √ tidyr   1.0.2     √ stringr 1.4.0
## √ readr   1.3.1     √ forcats 0.5.0
```

```
## Warning: package 'tibble' was built under R version 3.6.3
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

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

```
library(gridExtra)  # plot arrangement
```

```
## Warning: package 'gridExtra' was built under R version 3.6.3
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(ggplot2)
library(kernlab)      # SVM methodology
```

```
##
## Attaching package: 'kernlab'
```

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

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(e1071)        # SVM methodology
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
library(cluster)    # clustering algorithms
```

```
## Warning: package 'cluster' was built under R version 3.6.3
```

```
library(factoextra) # clustering algorithms & visualization
```

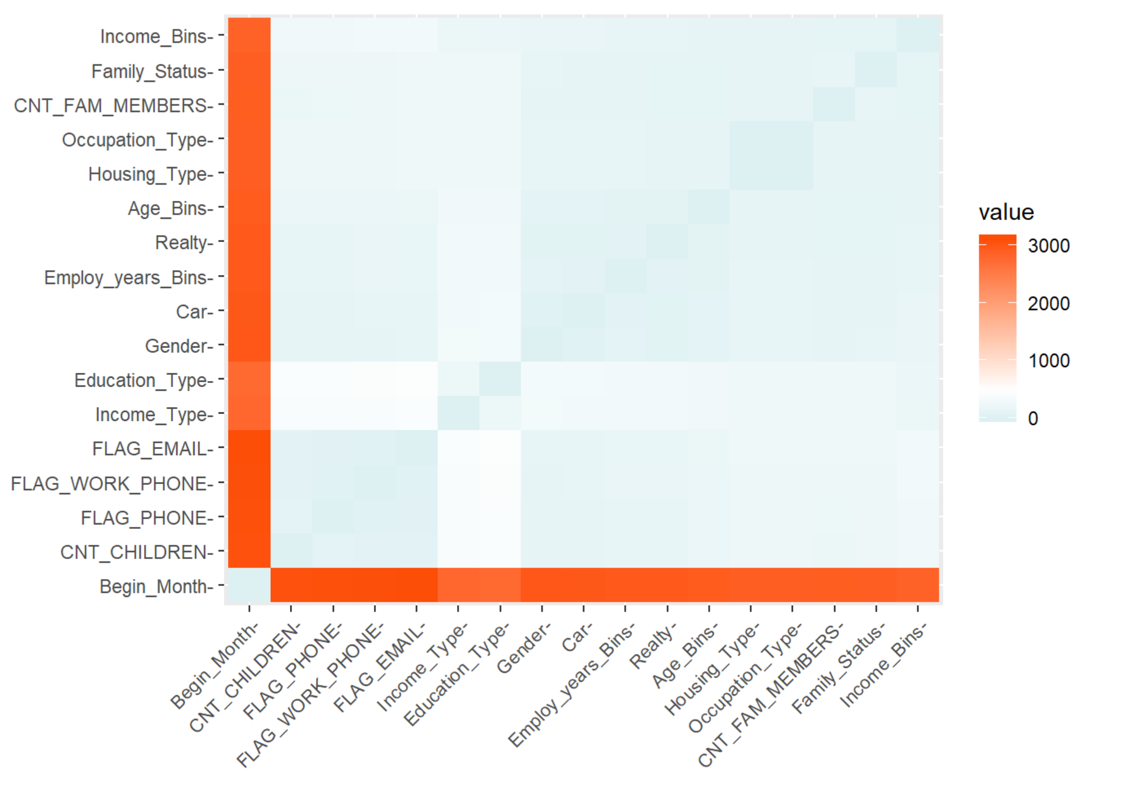
```
## Warning: package 'factoextra' was built under R version 3.6.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

K-means

```
use <- data[, c('CNT_CHILDREN','FLAG_WORK_PHONE','FLAG_PHONE','FLAG_EMAIL','CNT_FAM_MEMBERS','Begin_Month','Gende
r','Car','Realty','Income_Type','Family_Status','Housing_Type','Occupation_Type','Income_Bins','Age_Bins','Employ
_years_Bins','Education_Type')]
y <- data[,'Risk']
```

```
smp_size <- floor(0.5 * nrow(use))
sample_ <- sample(1:length(y),size = smp_size)
#use_ <- use[sample_,c('Income_Type','Family_Status','Begin_Month','Occupation_Type','Housing_Type')]
use_ <- use[sample_,]
distance <- get_dist(t(use_))
fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))
```




```
km.out=kmeans(use,2,nstart=15)
cluster_ <- km.out$cluster - 1

# Precision
precision <- length(cluster_[cluster_*y == 1])/(length(cluster_[cluster_-y == 1])+length(cluster_[cluster_*y == 1
]))
precision
```

```
## [1] 0.4552146
```

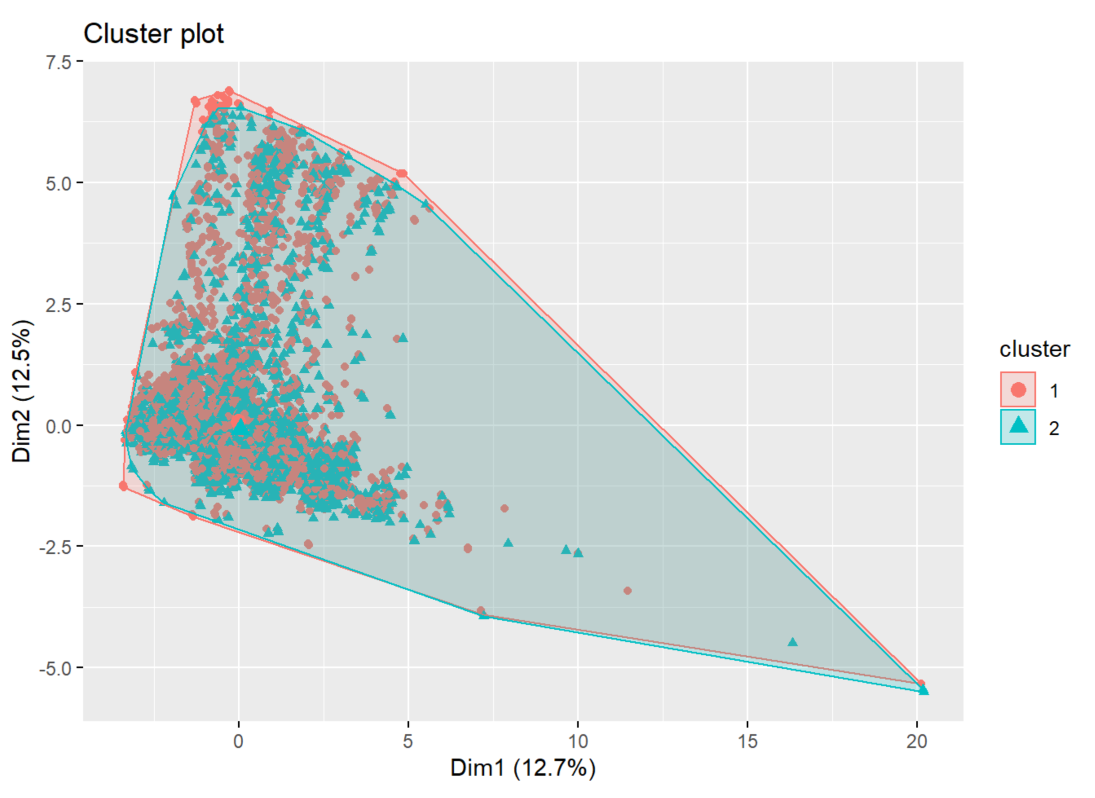
```
# Recall
recall <- length(cluster_[cluster_*y == 1])/(length(cluster_[cluster_-y == 1])+length(cluster_[cluster_*y == -1
]))
recall
```

```
## [1] 0.8355852
```

```
km.out$centers
```

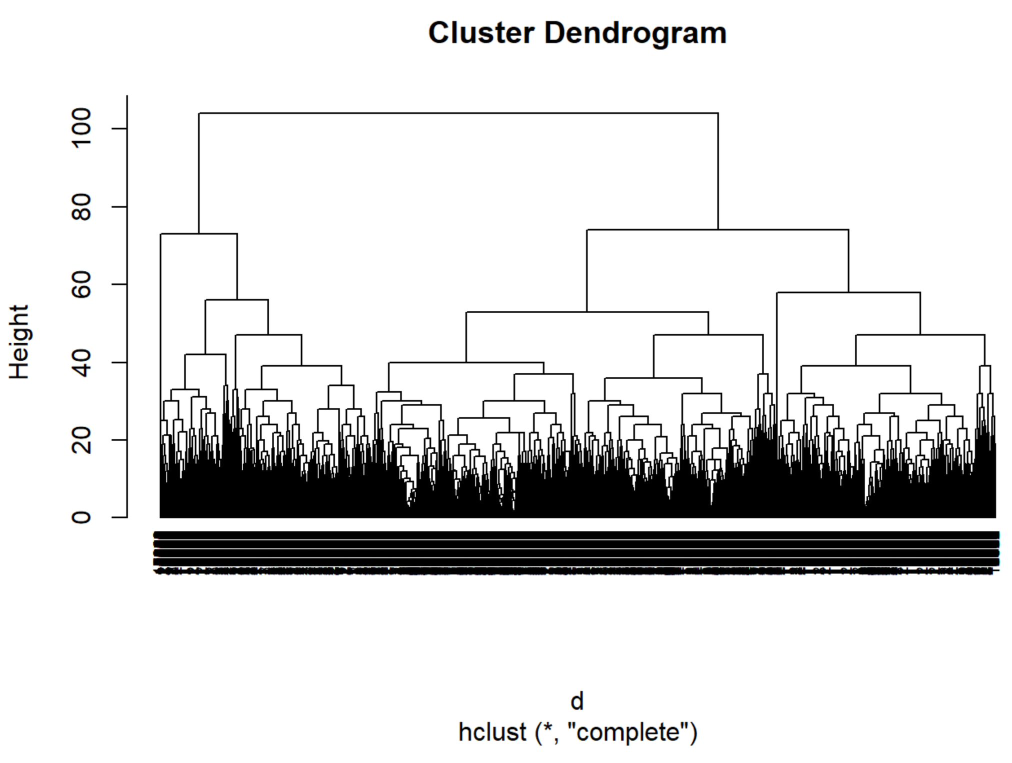
```
##   CNT_CHILDREN FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL CNT_FAM_MEMBERS
## 1    0.4328588       0.2289701  0.3584082 0.09861483        2.228272
## 2    0.4355312       0.2322480  0.3619631 0.09120983        2.267432
##   Begin_Month   Gender      Car   Realty Income_Type Family_Status Housing_Type
## 1    15.17483 1.379498 1.411644 1.680733    3.648549      2.306795     2.315027
## 2    40.18138 1.328254 1.406175 1.643242    3.793629      2.243644     2.259298
##   Occupation_Type Income_Bins Age_Bins Employ_years_Bins Education_Type
## 1        2.315027    2.905183 1.911349          1.623254       4.161175
## 2        2.259298    2.911223 1.911060          1.626005       4.136469
```

```
fviz_cluster(km.out, geom = "point",data = use)
```



H-clust

```
d <- dist(use, method = "manhattan")
hc1 <- hclust(d, method = "complete" )
plot(hc1, cex = 0.6, hang = -1)
```




```
sub_grp <- cutree(hc1, k = 2)
table(sub_grp)
```

```
## sub_grp
##     1     2
## 15447  5252
```

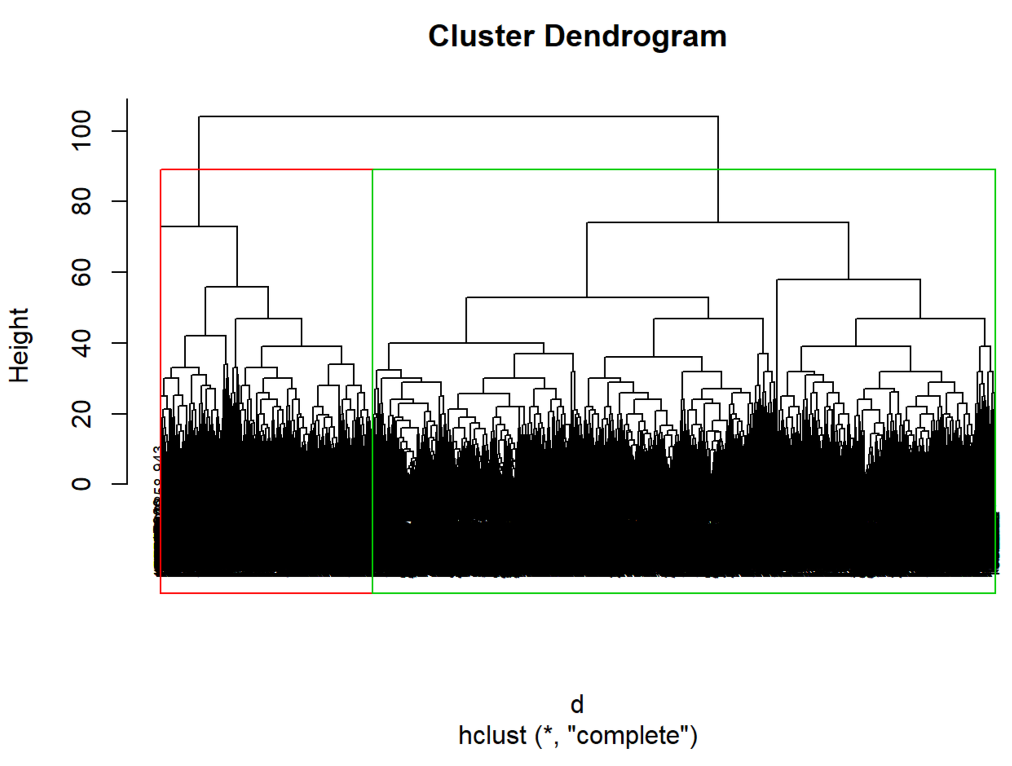
```
result = sub_grp-1

# Precision
precision <- length(result[result*y == 1])/(length(result[result-y == 1])+length(result[result*y == 1]))
precision
```

```
## [1] 0.3461538
```

```
# Recall
recall <- length(result[result*y == 1])/(length(result[result-y == 1])+length(result[result*y == -1]))
recall
```

```
## [1] 0.5294118
```

```
plot(hc1, cex = 0.6)
rect.hclust(hc1, k = 2, border = 2:5)
```



Final Project

```r
library("glmnet")
library("factoextra")
library("leaps")
library("gam")
library("precrec")
library("ModelMetrics")
```

PCA

```r
df = read.csv("CleanV3.csv")
df = subset(df, select= -c(ID, CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY, NAME_INCOME_TYPE, NAME_EDUCATION_TYPE,
NAME_FAMILY_STATUS, NAME_HOUSING_TYPE, FLAG_MOBIL, OCCUPATION_TYPE, Occupation_Type, Income_Bins, Age_Bins, Emplo
y_years_Bins))
y = df$Risk
table(y)
```

```
## y
##     0     1
## 11828  8871
```

```r
X.model = model.matrix(Risk ~ .-1, data = df)
head(X.model)
```

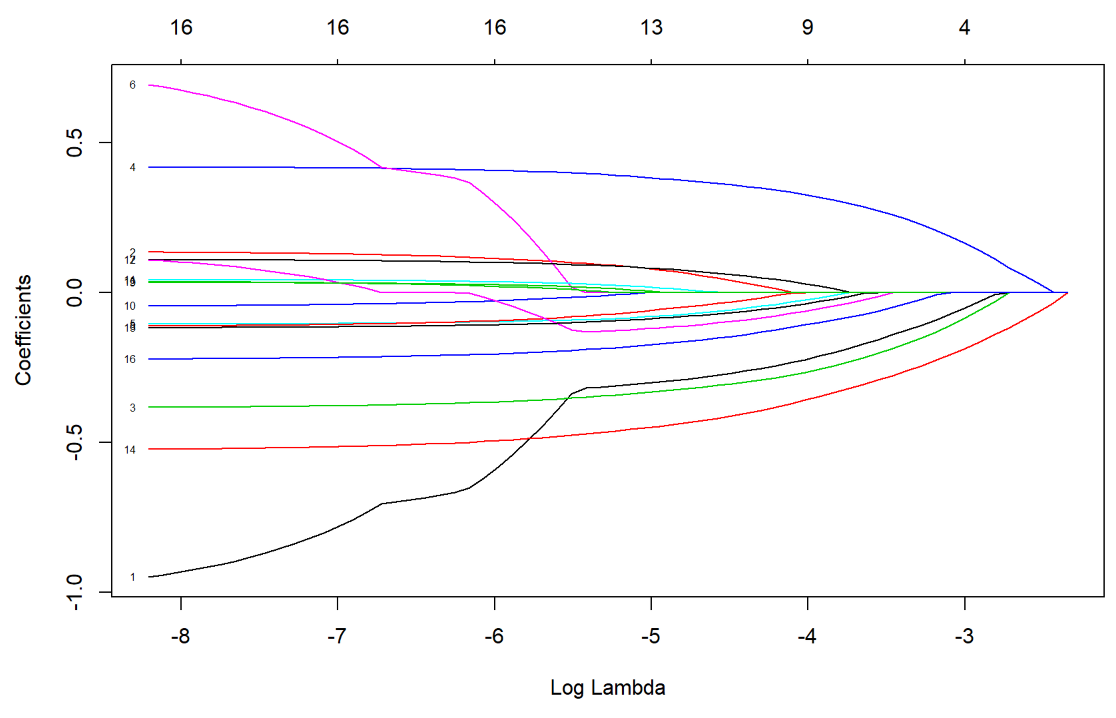
```
##   CNT_CHILDREN AMT_INCOME_TOTAL FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
## 1            0           270000               0          0          0
## 2            0           202500               0          0          0
## 3            0           157500               0          1          0
## 4            1           148500               0          0          0
## 5            0           180000               0          0          0
## 6            0           202500               0          0          0
##   CNT_FAM_MEMBERS Begin_Month Gender Car Realty Income_Type Family_Status
## 1               2           6      2   2      2           5             2
## 2               2           0      1   1      2           5             2
## 3               2          17      1   1      1           3             1
## 4               3          14      2   2      2           5             2
## 5               2          21      2   2      2           5             2
## 6               1          32      1   2      2           5             5
##   Housing_Type Age Employ_years Education_Type
## 1            2  39           11              2
## 2            2  56           10              2
## 3            3  50            8              4
## 4            2  41           14              2
## 5            2  59            7              2
## 6            2  33            4              4
```

```r
X.std = scale(X.model)
head(X.std)
```

```
##   CNT_CHILDREN AMT_INCOME_TOTAL FLAG_WORK_PHONE FLAG_PHONE FLAG_EMAIL
## 1   -0.5858234       0.73366355      -0.5694531 -0.8004671 -0.3342278
## 2   -0.5858234       0.06006953      -0.5694531 -0.8004671 -0.3342278
## 3   -0.5858234      -0.38899315      -0.5694531  1.4361611 -0.3342278
## 4    0.7571539      -0.47880568      -0.5694531 -0.8004671 -0.3342278
## 5   -0.5858234      -0.16446181      -0.5694531 -0.8004671 -0.3342278
## 6   -0.5858234       0.06006953      -0.5694531 -0.8004671 -0.3342278
##   CNT_FAM_MEMBERS Begin_Month     Gender        Car     Realty Income_Type
## 1      -0.2862711  -1.4040077  1.4177210  1.2673667  0.7462296   0.7586914
## 2      -0.2862711  -1.8070025 -0.7774155 -0.8712062  0.7462296   0.7586914
## 3      -0.2862711  -0.6651838 -0.7774155 -0.8712062 -1.4773941  -0.4331775
## 4       0.8564825  -0.8666812  1.4177210  1.2673667  0.7462296   0.7586914
## 5      -0.2862711  -0.3965206  1.4177210  1.2673667  0.7462296   0.7586914
## 6      -1.4290247   0.3423033 -0.7774155  1.2673667  0.7462296   0.7586914
##   Family_Status Housing_Type         Age Employ_years Education_Type
## 1    -0.3389826   -0.3123778 -0.01771028   0.64018424     -0.6276262
## 2    -0.3389826   -0.3123778  1.81126485   0.47217039     -0.6276262
## 3    -1.5763680    0.7515577  1.16574422   0.13614269      1.6751719
## 4    -0.3389826   -0.3123778  0.19746327   1.14422578     -0.6276262
## 5    -0.3389826   -0.3123778  2.13402517  -0.03187115     -0.6276262
## 6     3.3731735   -0.3123778 -0.66323091  -0.53591270      1.6751719
```

Feature Selection - LASSO

```r
fit.lasso = glmnet(X.std, y, family = "binomial", alpha = 1)
plot(fit.lasso, xvar="lambda", label=TRUE)
```

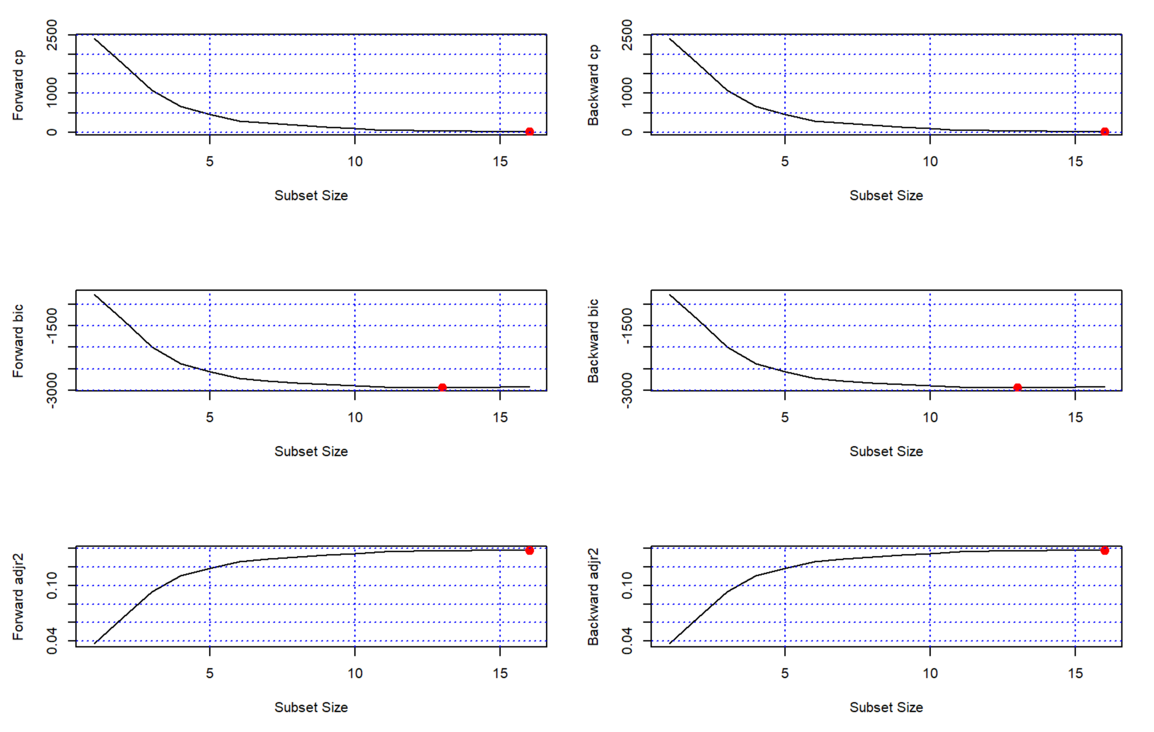



```r
coef(fit.lasso, exp(-3.5))
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)      -0.300569823
## CNT_CHILDREN     -0.147307494
## AMT_INCOME_TOTAL  .
## FLAG_WORK_PHONE  -0.193427408
## FLAG_PHONE        0.265634264
## FLAG_EMAIL        .
## CNT_FAM_MEMBERS   .
## Begin_Month       .
## Gender            .
## Car               .
## Realty            .
## Income_Type       .
## Family_Status    -0.005461668
## Housing_Type      .
## Age              -0.283246440
## Employ_years      .
## Education_Type   -0.054691948
```

Feature Selection - Forward Selection/Backward Selection

```r
fwd = regsubsets(X.model, y, nvmax = 16, method = "forward")
bwd = regsubsets(X.model, y, nvmax = 16, method = "backward")
fwd.summary = summary(fwd)
bwd.summary = summary(bwd)
par(mfrow = c(3, 2))
plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward cp", type = "l")
points(which.min(fwd.summary$cp), fwd.summary$cp[which.min(fwd.summary$cp)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward cp", type = "l")
points(which.min(bwd.summary$cp), bwd.summary$cp[which.min(bwd.summary$cp)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
plot(fwd.summary$bic, xlab = "Subset Size", ylab = "Forward bic", type = "l")
points(which.min(fwd.summary$bic), fwd.summary$bic[which.min(fwd.summary$bic)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
plot(bwd.summary$bic, xlab = "Subset Size", ylab = "Backward bic", type = "l")
points(which.min(bwd.summary$bic), bwd.summary$bic[which.min(bwd.summary$bic)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward adjr2", type = "l")
points(which.max(fwd.summary$adjr2), fwd.summary$adjr2[which.max(fwd.summary$adjr2)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward adjr2", type = "l")
points(which.max(bwd.summary$adjr2), bwd.summary$adjr2[which.max(bwd.summary$adjr2)],
col = 2, cex = 2, pch = 20)
grid(col = 4)
```




```r
fwd = regsubsets(X.std, y, nvmax = 6, method = "forward")
fwd.summary = summary(fwd)
coef(fwd, which.min(fwd.summary$bic))
```

```
##     (Intercept)    CNT_CHILDREN FLAG_WORK_PHONE      FLAG_PHONE
##      0.42857143     -0.16033583     -0.08198309      0.09611223
## CNT_FAM_MEMBERS             Age  Education_Type
##      0.10579632     -0.10021151     -0.04240820
```

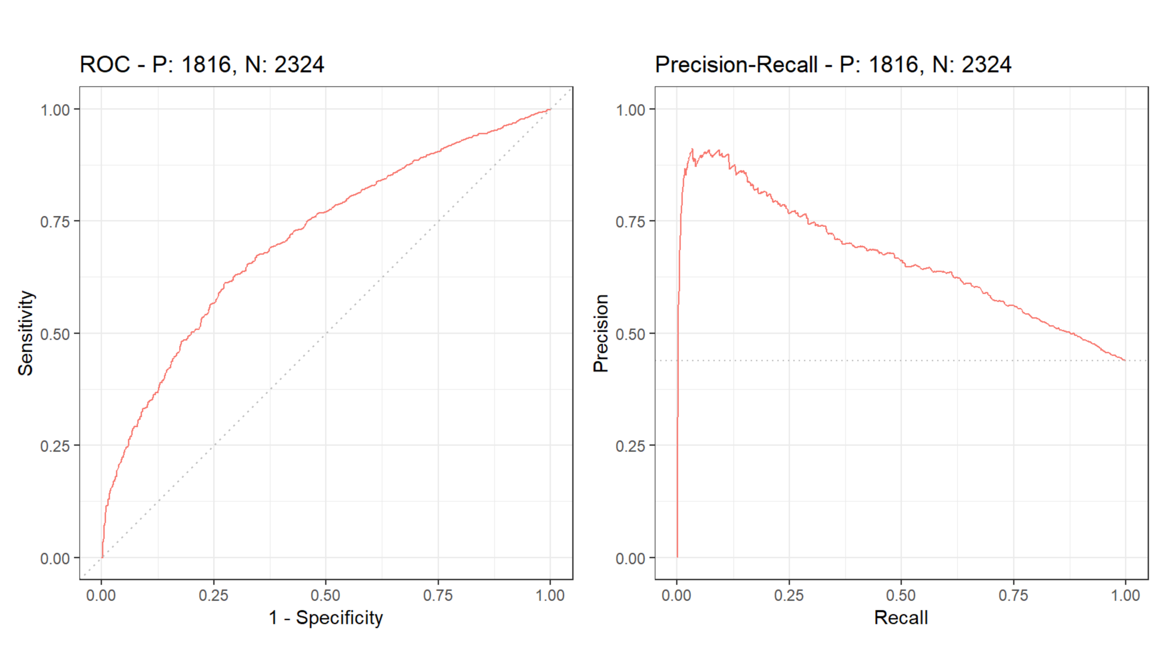
Logistic Regression

```r
set.seed(100)
train = sample(nrow(df), 0.8*nrow(df), replace = FALSE)
X_train = df[train,]
X_test = df[-train,]
y_train = y[train]
y_test = y[-train]
```

```r
# with selection
lr = glm(Risk ~ CNT_CHILDREN + FLAG_WORK_PHONE + FLAG_PHONE + Family_Status + Age + Education_Type, data = X_trai
n, family = "binomial")
summary(lr)
```

```
##
## Call:
## glm(formula = Risk ~ CNT_CHILDREN + FLAG_WORK_PHONE + FLAG_PHONE +
##     Family_Status + Age + Education_Type, family = "binomial",
##     data = X_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0277  -0.9817  -0.6703   1.1027   3.7627
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      2.69080    0.11138  24.159   <2e-16 ***
## CNT_CHILDREN    -0.45058    0.02602 -17.317   <2e-16 ***
## FLAG_WORK_PHONE -0.99548    0.04571 -21.776   <2e-16 ***
## FLAG_PHONE       0.98741    0.03993  24.727   <2e-16 ***
## Family_Status   -0.19087    0.02142  -8.911   <2e-16 ***
## Age             -0.04979    0.00191 -26.064   <2e-16 ***
## Education_Type  -0.22425    0.01977 -11.340   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22592  on 16558  degrees of freedom
## Residual deviance: 20431  on 16552  degrees of freedom
## AIC: 20445
##
## Number of Fisher Scoring iterations: 4
```

```r
y_pred = predict(lr, X_test, type="response")
precrec_obj = evalmod(scores = y_pred, labels = y_test)
autoplot(precrec_obj)
```




```r
recall(y_test, y_pred, cutoff = 0.4)
```

```
## [1] 0.6993392
```

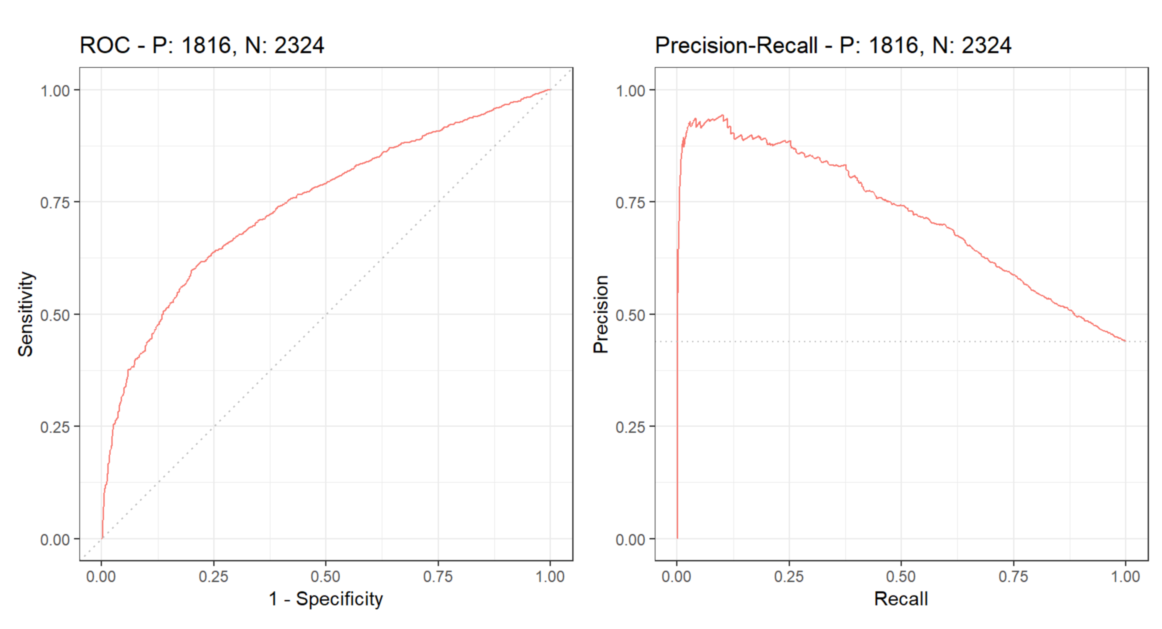
```r
precision(y_test, y_pred, cutoff = 0.4)
```

```
## [1] 0.5788514
```

```r
# generalized additive model (with Smoothing Splines)
lr_gam = gam(Risk ~ s(CNT_CHILDREN) + FLAG_WORK_PHONE + FLAG_PHONE + s(Family_Status) + s(Age) + s(Education_Typ
e), data = X_train, family = "binomial")
summary(lr_gam)
```

```
##
## Call: gam(formula = Risk ~ s(CNT_CHILDREN) + FLAG_WORK_PHONE + FLAG_PHONE +
##     s(Family_Status) + s(Age) + s(Education_Type), family = "binomial",
##     data = X_train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1549 -0.9412 -0.6120  1.0064  2.6832
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 22592.12 on 16558 degrees of freedom
## Residual Deviance: 19456.46 on 16540 degrees of freedom
## AIC: 19494.46
##
## Number of Local Scoring Iterations: 5
##
## Anova for Parametric Effects
##                     Df  Sum Sq Mean Sq F value    Pr(>F)
## s(CNT_CHILDREN)      1   213.3  213.29 211.727 < 2.2e-16 ***
## FLAG_WORK_PHONE      1   186.0  185.97 184.604 < 2.2e-16 ***
## FLAG_PHONE           1   536.9  536.88 532.947 < 2.2e-16 ***
## s(Family_Status)     1    47.9   47.92  47.572 5.494e-12 ***
## s(Age)               1   560.7  560.67 556.568 < 2.2e-16 ***
## s(Education_Type)    1   128.2  128.24 127.303 < 2.2e-16 ***
## Residuals        16540 16662.0    1.01
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                   Npar Df Npar Chisq    P(Chi)
## (Intercept)
## s(CNT_CHILDREN)         3     218.70 < 2.2e-16 ***
## FLAG_WORK_PHONE
## FLAG_PHONE
## s(Family_Status)        3     369.90 < 2.2e-16 ***
## s(Age)                  3      95.38 < 2.2e-16 ***
## s(Education_Type)       3     241.95 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
y_pred_gam = predict(lr_gam, X_test, type="response")
precrec_obj_gam = evalmod(scores = y_pred_gam, labels = y_test)
autoplot(precrec_obj_gam)
```




```r
recall(y_test, y_pred_gam, cutoff = 0.4)
```

```
## [1] 0.7120044
```

```r
precision(y_test, y_pred_gam, cutoff = 0.4)
```

```
## [1] 0.6050538
```