ECON-4280

Credit Card Approval Analysis

Xiaowen Zhu

1. Introduction

The decision to approve a credit card depends primarily on the applicant's personal and financial background. Factors such as age, gender, income, employment status, credit history, and other attributes are all important in approving decisions. Credit analysis involves a measure that investigates the possibility of a third party repaying a bank on time and predicts its default characteristics. Therefore, the assessment of credit approval is important before jumping to any grant decision.

2. Data

2.1 Data description

The "Credit Approval Data Set" is downloaded online from UCI Machine Learning Repository. The dataset includes 690 rows and 16 columns. Each row indicates an applicant and the first fifteen columns each indicates an independent variable that applicants have. The last column is the results of credit approval, "+" means approved while "-" means not approved. The data says about 44.5% of applicants received approval while 55.5% failed to get the credit card.

2.2 Data preprocessing

After reviewing the data, we can easily find that about 5% of rows (37 cases) have missing values that is shown as "?" in the dataset. The missing values mostly appear in attributes of Gender, Age, MaritalStatus, BankCustomer, EducationLevel, Ethinicity, and Zipcode. For the reasons that the number of missing values is not very large, and the method of filling the mean value into empty blocks may have large deviation and thus distort the final results, I choose to just delete those thirty-seven rows. After data cleaning process, the row with missing values are deleted and the number of entries decreases from 690 to 653.

The type of attributes also needs to be transformed. After the employment of functions as.numeric() and as.factor() in R Studio, the attributes Age, Debt, YearsEmployed, CreditScore, and Income are transformed from character to numeric, and the attributes MaritalStatus, DriverLicense, Citizen, and Zipcode are transformed from character to factor. Then for binary factors, set male/t/approved/+ for 1 and female/f/not approved/- for 0.

Normalization is a crucial part for data cleaning. Originally, the scale of each feature is different and too centered. From Figure 1 (before normalization) that shown below, we can find that each data points and distributions are hard to see and distinguish.

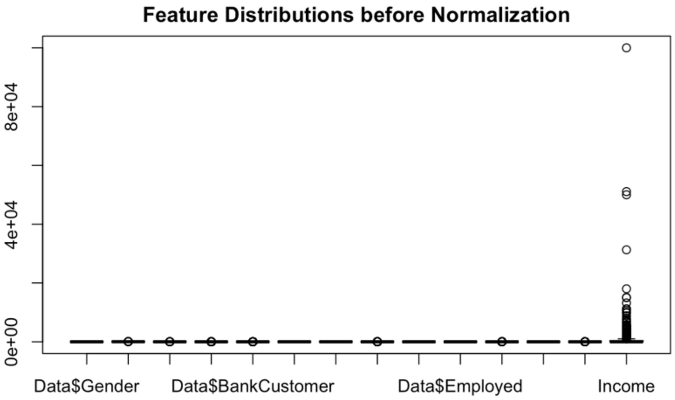


Figure 1

The function "scale" calculates the mean and standard deviation of the entire vector, and then "scale" each element by those values by subtracting the mean and dividing by the standard deviation, which provides standardization for each feature. According to Figure 2 (with normalization), the mean and standard deviation of each feature are clear, and the trend of each feature is easily defined.

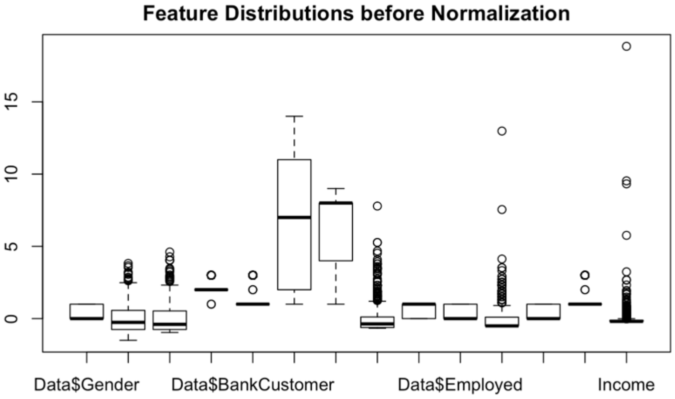


Figure 2

At this time, the dataset is ready for further analysis and visualization.

2.3 Data Separation

For better and more accurate analysis, I separated the data into training and testing sets with the ratio of 8:2. By thinking that some attributes like Zipcode does not have an influence on the final decision of credit card approval or not, I chose not to include them in the following models. As a result, only nine attributes (Gender, Age, Debt, YearsEmployed, PriorDefault, Employed, CreditScore, DriversLicense, and Income) are considered for further analysis.

3. Data Analysis

3.1 PCA Analysis

PCA analysis is an effective method to extract important features from a large amount of data (TEAM, 2016). It also reduces the dimension of the data set with a motive to capture as much information as possible. A principal component is a normalized linear combination of the original perdition of the data set. The scree plot provides the number of principal components that should be selected for modeling. It is used to access components or factors which explains the most variability in the data. According to the Figure 3, the "elbow" occurs around x=4 to x=6. We should pick the number of principal components in this range.

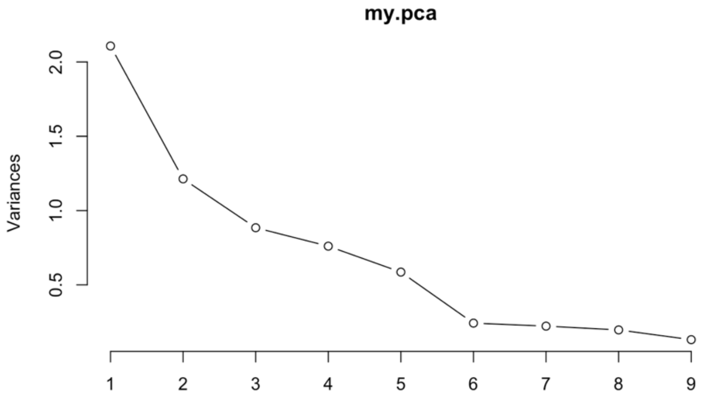


Figure 3

According to the PCA biplot (Figure 4), we can say the vectors of feature "Income" and "YearsEmployed" are longer than others. So, we can say there are important features that will affect predictions positively. In addition, the smaller angle between two features, the closer the inner relationship between these two features. For example, Debt, Age, YearsEmployed and other features are too close to be distinguished. Eventually, we need to discard some low-impact features that can be found in the following part of the paper.

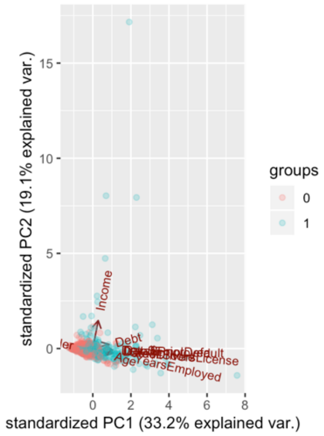


Figure 4

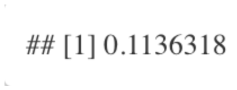
3.2 Feature Selection

3.2.1 Model 1: Linear Regression Model

I first fit the dataset with linear regression model.

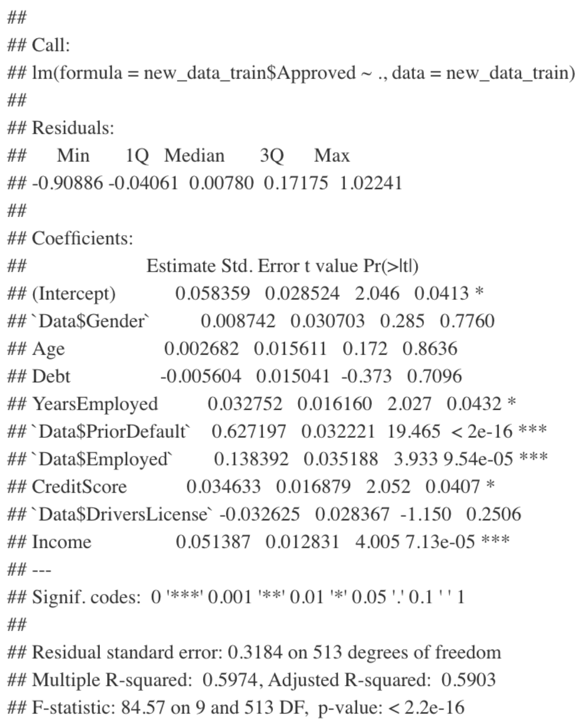
We can consider a linear model to be statistically significant only when all these p-values are less than the pre-determined statistical significance level of 0.05. This can visually be interpreted by the significance stars at the end of the row against each X variables. The more the stars beside the variable’s p-value, the more significant that variable is.

According to the summary below, it can be easily found that PriorDefault, Employed, and Income are the most significant variables, followed by CreditScore and YearsEmployed. The test error is about 0.114.



```
## [1] 0.1136318
```

Figure 5 Test error of Linear Regression Model

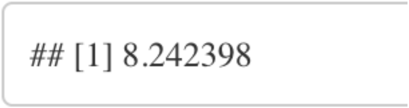


```
##
## Call:
## lm(formula = new_data_train$Approved ~ ., data = new_data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90886 -0.04061  0.00780  0.17175  1.02241
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.058359   0.028524   2.046   0.0413 *
## `Data$Gender`          0.008742   0.030703   0.285   0.7760
## Age                    0.002682   0.015611   0.172   0.8636
## Debt                  -0.005604   0.015041  -0.373   0.7096
## YearsEmployed          0.032752   0.016160   2.027   0.0432 *
## `Data$PriorDefault`    0.627197   0.032221  19.465  < 2e-16 ***
## `Data$Employed`        0.138392   0.035188   3.933 9.54e-05 ***
## CreditScore            0.034633   0.016879   2.052   0.0407 *
## `Data$DriversLicense` -0.032625   0.028367  -1.150   0.2506
## Income                 0.051387   0.012831   4.005 7.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3184 on 513 degrees of freedom
## Multiple R-squared:  0.5974, Adjusted R-squared:  0.5903
## F-statistic: 84.57 on 9 and 513 DF,  p-value: < 2.2e-16
```

Figure 6 Summary of Linear Regression Model

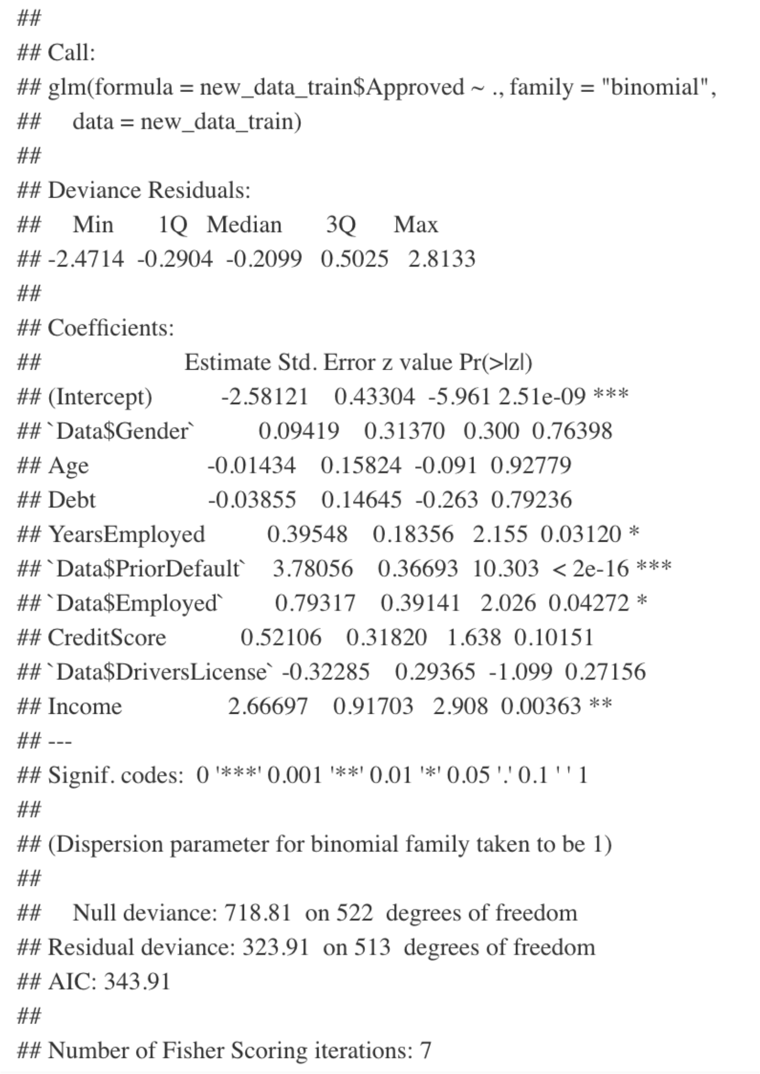
3.2.2 Model 2: Logistic Regression Model

For the second model I choose Logistic Regression Model. The same as what I've done for the previous model, I fit the dataset with glm() function and calculated the test error. From Figure 7 and Figure 8 below, we can see the error is about 8.24 and the results showed in summary sheet is pretty similar to that of Linear Regression Model. By find p-values that are less than 0.05, we can know YearsEmployed, PriorDefault, Employed and Income have significant relationship with the dependent variable “approved”.



```
## [1] 8.242398
```

Figure 7 Test error of Logistic Regression Model

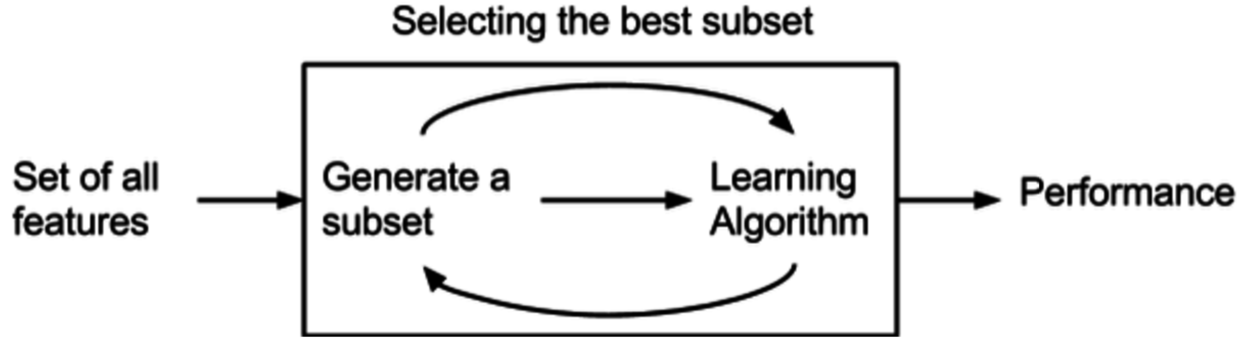


```
##
## Call:
## glm(formula = new_data_train$Approved ~ ., family = "binomial",
##     data = new_data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4714  -0.2904  -0.2099   0.5025   2.8133
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -2.58121    0.43304  -5.961 2.51e-09 ***
## `Data$Gender`           0.09419    0.31370   0.300  0.76398
## Age                    -0.01434    0.15824  -0.091  0.92779
## Debt                   -0.03855    0.14645  -0.263  0.79236
## YearsEmployed           0.39548    0.18356   2.155  0.03120 *
## `Data$PriorDefault`     3.78056    0.36693  10.303  < 2e-16 ***
## `Data$Employed`         0.79317    0.39141   2.026  0.04272 *
## CreditScore             0.52106    0.31820   1.638  0.10151
## `Data$DriversLicense`  -0.32285    0.29365  -1.099  0.27156
## Income                  2.66697    0.91703   2.908  0.00363 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 718.81  on 522  degrees of freedom
## Residual deviance: 323.91  on 513  degrees of freedom
## AIC: 343.91
##
## Number of Fisher Scoring iterations: 7
```

Figure 8 Summary of Logistic Regression Model

3.2.3 Model 3: Wrapper Method for Feature Selection

The basic idea of Wrapper method is to examine all possible combinations of the features that can be computationally very expensive, particularly if the feature set is very large (Malik, 2018). Wrapper method starts with an empty feature subset and adds one feature at the time in each round. The added feature is selected from the pool of all features that are not in the feature subset, and it is the feature that – when added – results in the best classifier performance.



Resource: (KAUSHIK, 2016)

According to the learning algorithm of Wrapper method, I used "for loop" to test all combinations of feature subsets and returned the corresponding AUC accuracy. According to the graph below, the AUC accuracies are quite high and similar for all combinations of feature subsets, due to limited numbers of features. However, it can still distinguish that 4 of feature selected generates highest AUC accuracy, around 95%. Then I pick 4 features because these features show the most important (largest weight) that can affect our predictions positively to a large extent. Then I rerun the loop to create a new internal training data set with 4 selected features.

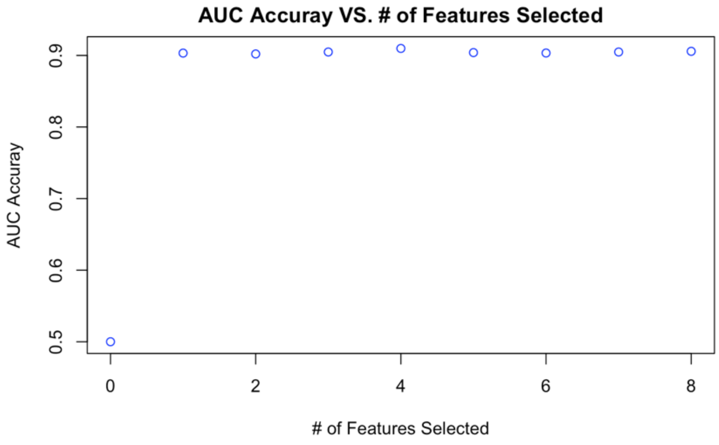


Figure 9

Below is the PCA analysis after feature selection. PCA analysis is statistic method to analyze the structure of a correlation matrix. By construction, the first principal component is the one that maximizes the variance when data are projected onto a line. The second principal component is orthogonal to the first one and still maximizes the remaining variance. In general, it assumes that two components explain a sufficient amount of the variance to provide a meaningful visual representation of the structure of cases and variables.

We can look to see which features are close together in the space. Where this applies, this may suggest that applicants who are good at one condition/requirement are likely also to be good at the other condition/requirement. Alternatively, we can use the plot to see which features are distant. For example, applicants who have a high credit score are more likely to have long years employed.

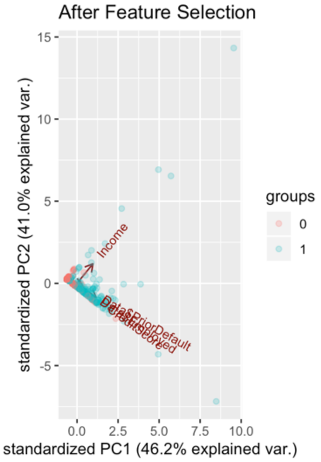


Figure 10

One thing I need to mention is that Wrapper method has an overfitting problem. Overfitting occurs when Wrapper method finds a feature subset that has a high internal accuracy while generalizing poorly (Loughrey & Cunningham). However, a large dataset with several features is more likely to have this problem. Since only having 9 features in the small dataset, I do not define that it would be a problem in this case, in which our feature selection result will not be impacted to a large extent.

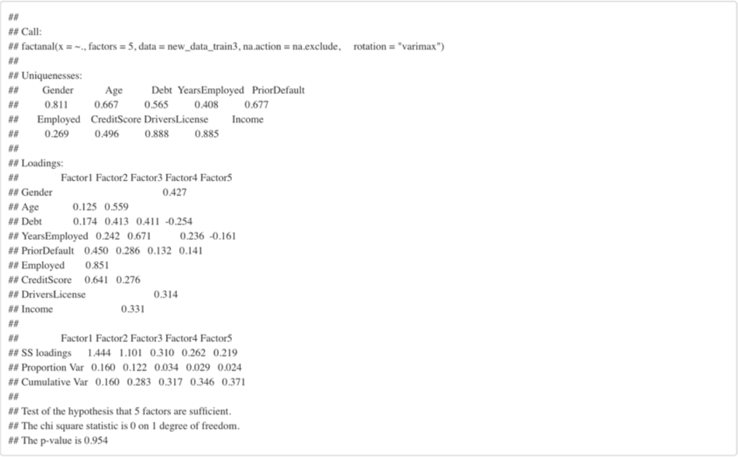
Compared with other feature selection model, Wrapper methods tests all possibilities of feature subsets, which generates more specific and reliable analysis. As a result, I pick it as my final feature selection model.

3.2.4 Model 4: Factor Analysis for Feature Selection

Factor analysis is a statistic method to reduce the dimensions of the database. It is able to reduce all 9 features into few factor groups. 9 features will be divided into a few factor groups based on their inner correlations. With only a few factor groups. it is easier for us to find the correlations of the approval of credit cards.

According to the previous PCA graph, the "elbow" of the line around x=2 to x=6. Theoretically, I need to pick factor number as 6 because a suitable but larger factor number can provide a more detailed analysis. However, since we only have 9 features in the dataset, 6 factor groups will be too much. Therefore, we pick the factor number as 5.

Below is the factor analysis with 5 factor groups. First, we can look at the sums of squared (SS) loadings; these are the eigenvalues, or the variance in all variables which is accounted for by that factor (i.e., the eigenvalue/# of variables = proportion variance). If a factor has a “high” SS loading (i.e., eigenvalue), then it is helping to explain the variances in the variables. In the “factanal()” output, the factors are ordered by their eigenvalues, with higher eigenvalues first. A general rule is to choose the eigenvalue larger than 1. Here, factor 1 and factor 2 appear to be more important.



```
##
## Call:
## factanal(x = ~., factors = 5, data = new_data_train3, na.action = na.exclude,     rotation = "varimax")
##
## Uniquenesses:
##        Gender           Age          Debt YearsEmployed  PriorDefault 
##         0.811         0.667         0.565         0.408         0.677 
##      Employed   CreditScore DriversLicense        Income 
##         0.269         0.496         0.888         0.885 
##
## Loadings:
##               Factor1 Factor2 Factor3 Factor4 Factor5
## Gender                                 0.427         
## Age            0.125   0.559                         
## Debt           0.174   0.413   0.411  -0.254         
## YearsEmployed  0.242   0.671           0.236  -0.161 
## PriorDefault   0.450   0.286   0.132   0.141         
## Employed       0.851                                 
## CreditScore    0.641   0.276                         
## DriversLicense                         0.314         
## Income                 0.331                         
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      1.444   1.101   0.310   0.262   0.219
## Proportion Var   0.160   0.122   0.034   0.029   0.024
## Cumulative Var   0.160   0.283   0.317   0.346   0.371
##
## Test of the hypothesis that 5 factors are sufficient.
## The chi square statistic is 0 on 1 degree of freedom.
## The p-value is 0.954
```

Figure 11

Looking at the reduced data by scatter plot, the distribution of each factor group and their inner relationships are shown by graphs and numbers respectively. For example, the distribution of factor group 1 is not normal and may impact the prediction model to a significant level. On the other hand, a positive number indicates a positive correlation, and larger numbers demonstrate a stronger relationship. For example, 0.67 implies that factor group1 and factor group 2 have a strong and positive correlation. In other words, applicants with a higher score in factor group 1 tend to have a higher score in factor group 2.

In addition, factor group 1, 2, and 4 have strong correlations with each other, while factor group 3 and 5 seems not to play an important role in the dataset. If we consider discarding some unimportant or low-impact features, factor group 3 and 5 should be taken into account first.

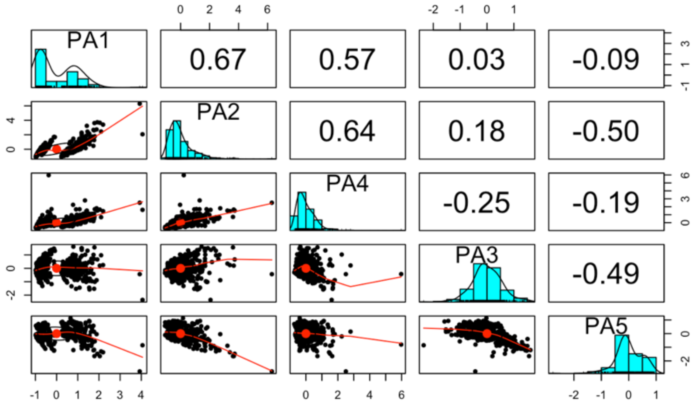


Figure 12

Based on their inner correlations, factor analysis divide employed, credit score and prior default into the same group. We can find a strong inner relationship in factor group 1. For example, applicants with a higher credit score are less likely to have more prior default. Similar in factor group 2, older applicants are more likely to have long years employed.

However, I decide not to go with factor analysis because of limited features and the difficulty to rescale factor groups. First of all, factor analysis is beneficial to reduce a large amount of data. But we only have 9 features which are already small data set. Keeping reducing data will distort the prediction model and results. Besides, future factor analysis requires rescaling all data in one factor group into one single column as a new feature. For example, features in factor group 1 all have different number scale and evaluation criteria. It is difficult to find a way to combine these three features.

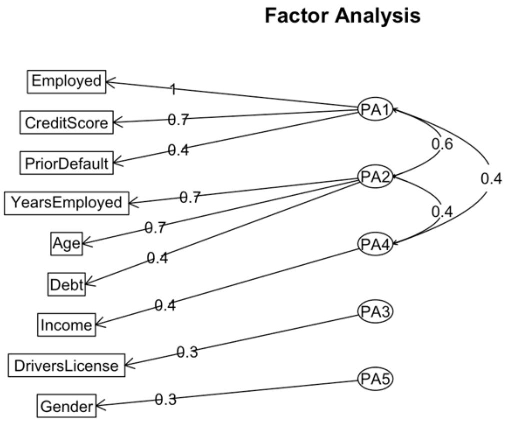
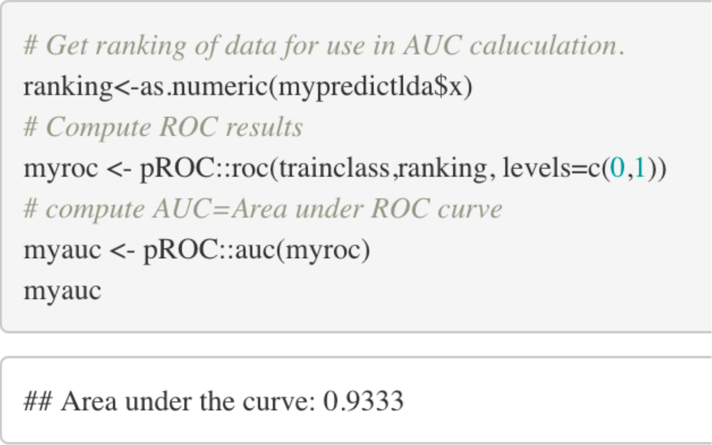


Figure 13

3.3 Prediction Model

3.3.1 Model 1: LDA Model

LDA is a linear transformation technique that are commonly used for dimensionality reduction. It is pretty much like PCA model, but it additionally finds the component axes that maximize the variance of our data. This makes a model of the distribution of predictors for each class and applies Bayes' theorem to turn them into predictions.

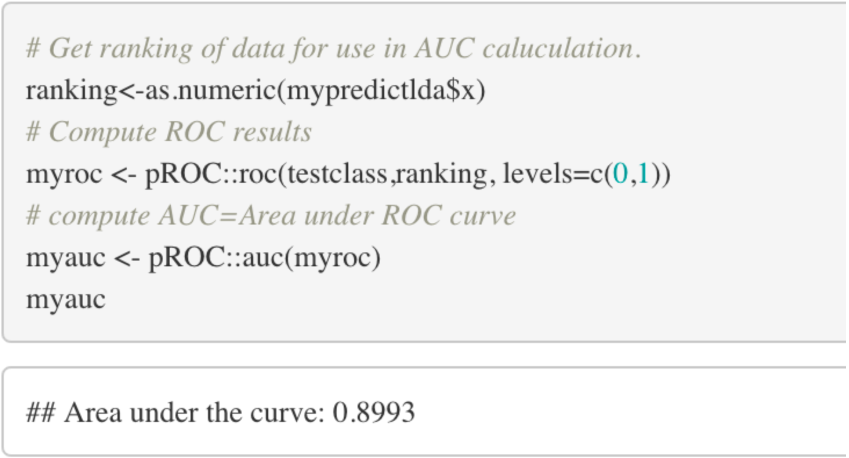


```
# Get ranking of data for use in AUC caluculation.
ranking<-as.numeric(mypredictlda$x)
# Compute ROC results
myroc <- pROC::roc(trainclass,ranking, levels=c(0,1))
# compute AUC=Area under ROC curve
myauc <- pROC::auc(myroc)
myauc
```

```
## Area under the curve: 0.9333
```

Figure 14 AUC of LDA Model (Train model through training dataset)

As Figure 14 shows, the AUC accuracy of training data is 93.33%, which is not bad. However, when I use the training model to predict test data set, I get 89.93% AUC accuracy (Figure 15), which decreases from previous one. In this case, LDA Model may not be suitable for the dataset.



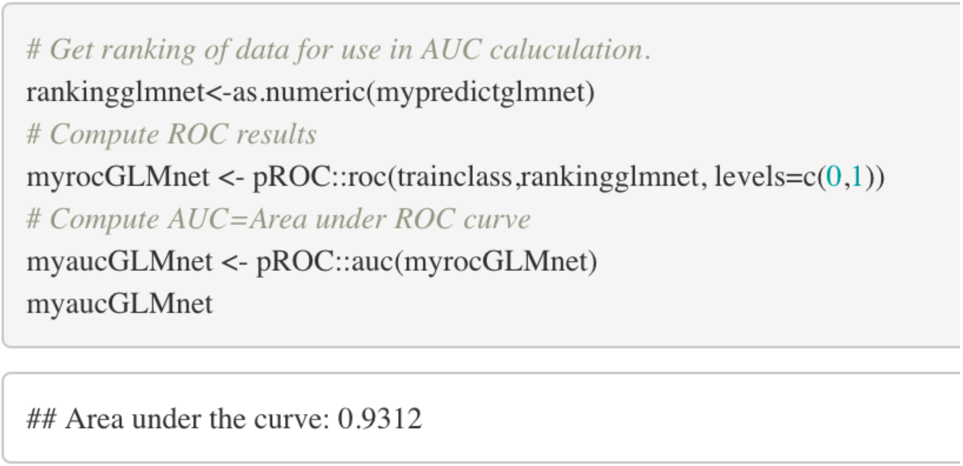
```
# Get ranking of data for use in AUC caluculation.
ranking<-as.numeric(mypredictlda$x)
# Compute ROC results
myroc <- pROC::roc(testclass,ranking, levels=c(0,1))
# compute AUC=Area under ROC curve
myauc <- pROC::auc(myroc)
myauc
```

```
## Area under the curve: 0.8993
```

Figure 15 AUC of LDA Model (Predict test dataset using training model)

3.3.2 Model 2: Logistic Regression Model with Lasso

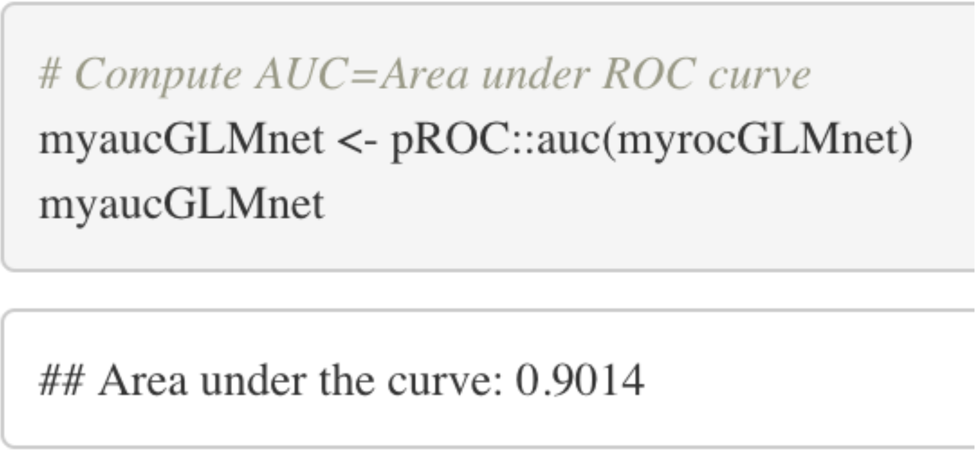
As what I’ve done in previous model, I first fit logistic regression model by glmnet() and predict training data with lasso by predict(). After that I calculate the area under ROC curve, which gives me an accuracy of 93.12%. But the predict test accuracy decreases to 90.14%, so I decide not to choose this model as well.



```
# Get ranking of data for use in AUC caluculation.
rankingglmnet<-as.numeric(mypredictglmnet)
# Compute ROC results
myrocGLMnet <- pROC::roc(trainclass,rankingglmnet, levels=c(0,1))
# Compute AUC=Area under ROC curve
myaucGLMnet <- pROC::auc(myrocGLMnet)
myaucGLMnet
```

```
## Area under the curve: 0.9312
```

Figure 16 AUC of Lasso Model (Train model through training dataset)



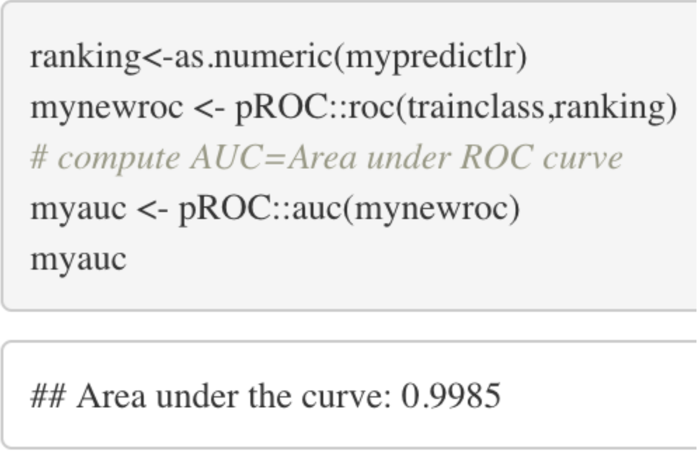
```
# Compute AUC=Area under ROC curve
myaucGLMnet <- pROC::auc(myrocGLMnet)
myaucGLMnet
```

```
## Area under the curve: 0.9014
```

Figure 17 AUC of Lasso Model (Predict test dataset using training model)

3.3.3 Model 3: Random Forest for Prediction

The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees, this model uses Random sampling of training data points when building trees (Koehrsen, 2018). Although each tree may have high variance respect to a particular subset of training data, the whole forest will have lower variance but not at the cost of increasing the bias. This is proved by the extremely high AUC accuracy of training data: 99.85%.

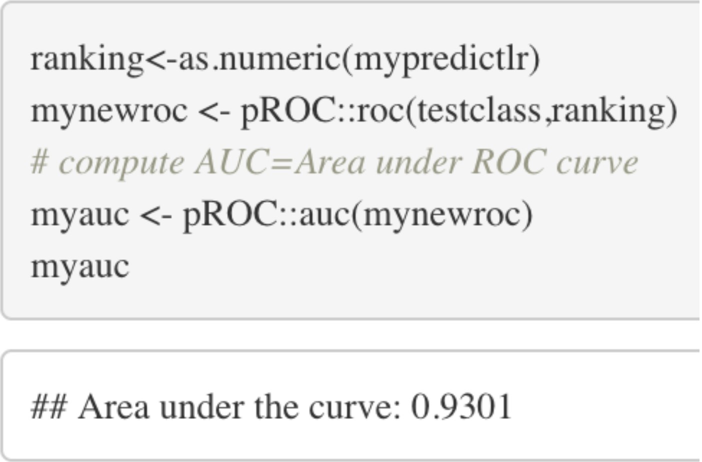


```
ranking<-as.numeric(mypredictlr)
mynewroc <- pROC::roc(trainclass,ranking)
# compute AUC=Area under ROC curve
myauc <- pROC::auc(mynewroc)
myauc
```

```
## Area under the curve: 0.9985
```

Figure 18 AUC of Random Forest (Train model through training dataset)

Using the training model to predict test data set, I get 93.01% AUC accuracy which is higher than that of other prediction models. High accuracy is not the only reason that I decide to go with Random Forest as a prediction model. Random Forest is a highly flexible model with high capability.



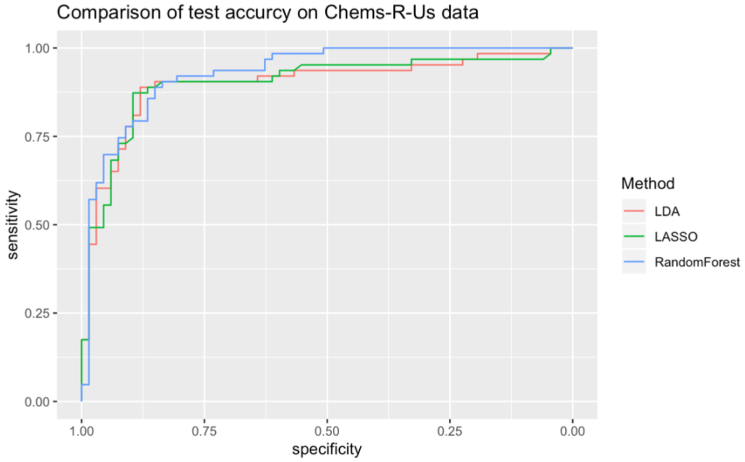
```
ranking<-as.numeric(mypredictlr)
mynewroc <- pROC::roc(testclass,ranking)
# compute AUC=Area under ROC curve
myauc <- pROC::auc(mynewroc)
myauc
```

```
## Area under the curve: 0.9301
```

Figure 19 AUC of Random Forest (Predict test dataset using training model)

3.4 Compare Three Prediction Model

Generally speaking, the line which is closest to the left and upper corner will have the highest accuracy and fitting model. Actually, three methods have similar logistics. All of them first to find the significance (coefficients) of each feature, and then calculate their AUC. And then based on their AUC, we decide to drop this feature or not. That's why the four lines in the graph are closed and show similar trends. More importantly, we only have 9 features to examine. The difference of three methods is hard to distinguish with a small data set.



Figure

4. Final Recommendation: Wrapper Method + Random Forest

In conclusion, based on my analysis above, I decide to utilize the combination of Wrapper method and random forest. I selected four features by Wrapper method: “PriorDefault”, “Employed”, “CreditScore”, and “Income”. The training model generates an AUC accuracy with 94.09%, while the predicting model generates 89.98% accuracy. Although the AUC accuracy is lower than simple Random Forest prediction mode with all features, Wrapper method provides a more detailed and reliable explanation for each feature. Discarding low-impact features is required for a large data set but may lower the accuracy in small data set (our case).

5. Additions

Since the aim of project is mainly to fit the dataset with different models and then pick the best, I put more words to emphasize on wrapper method and random forest rather than other models. If you want to see the entire results of models discussed above, I also have the code and corresponding results with pdf format (which I will submit along with this paper).

Reference

[1] KAUSHIK, S. (2016, December 1). *Introduction to Feature Selection methods with an example (or how to select the right variables?).* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/

[2] Koehrsen, W. (2018, August 30). *An Implementation and Explanation of the Random Forest in Python.* Retrieved from Towards: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[3] Loughrey, J., & Cunningham, P. (n.d.). *Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets*.*

[4] Malik, U. (2018, November 06). *Applying Wrapper Methods in Python for Feature Selection.* Retrieved from StackAbuse: https://stackabuse.com/applying-wrapper-methods-in-python-for-feature-selection/

[5] TEAM, A. V. (2016, March 21). *Practical Guide to Principal Component Analysis (PCA) in R & Python.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/