Loan Origination Prediction Based on Various Machine Learning Models

Jiajia Song[1] js5759, Yiyang Xu[1] yx2612, Xuhao Meng[1] xm2277, Yuchen Tang[1] yt2754, Yucen Wang[1] yw3561, Jia Qi Wang[1] jw4001

[1]Department of Statistics, Columbia University

1. Objective

The mortgage loan has always been a controversial topic in the economy because it plays such an essential part in our whole economy. As we all know, the great recession during 2007 and 2008 was led by the collapse of subprime mortgages. There were defaults on high-risk housing loans, which caused the credit crunch in the global banking system (Field Ann). We can see that if large defaults take place, it would disturb the economy badly. To avoid a bad impact on the economy by not lending money to people who are likely to default, banks may change their loan origination behavior pattern. We are in urgent need of an up-to-date prediction model. Therefore, in this project, we try to come up with a model to predict if people get rejected in a mortgage loan request by using the individual data in New York from Consumer Financial Protection Bureau (CFPB). We choose the year 2017 data because 2017 can be regarded as the middle of an economic boom according to the federal funds rate (graph above). The FED ended open market operations in 2014 and started raising interest rates from the end of 2015 in order to decrease the money supply in the nation's financial systems. This encouraged banks to lend money less easily, which increased the lending rates and mortgage rates (Thangavelu). We can notice that mortgage rates went up over time from the end of 2015 according to the federal funds rate. It will be harder for people to get a mortgage than it was before 2016. Hopefully, the results from our project may provide insights into the up-to-date institution's criterion. What's more, we will relate them to the corresponding mortgage policies and provide a broader view of the loan origination.

2. Data Description

2.1. *Variables*

Through implementing data validation and preprocessing on the raw dataset, we finally select nineteen variables to fit our models for further analysis. They may affect the loan origination process from different perspectives. For example, the income represents the applicants' direct loan repayment ability. Some variables related to the house location may influence the potential appreciation.

The description of the nineteen variables is listed in Table 1.

Table 1: Variables Description

Variables	**Descriptions**
Agency	1 – Office of the Comptroller of the Currency (OCC) 2 – Federal Reserve System (FRS) 3 – Federal Deposit Insurance Corporation (FDIC) 5 – National Credit Union Administration (NCUA) 7 – Department of Housing and Urban Development (HUD) 9 – Consumer Financial Protection Bureau (CFPB)
Loan Type	1 – Conventional (any loan other than FHA, VA, FSA, RHS loans) 2 – FHA-insured (Federal Housing Administration) 3 – VA-guaranteed (Veterans Administration) 4 – FSA/RHS (Farm Service Agency or Rural Housing Service)
Property Type	1 – One to four-family (other than manufactured housing) 2 – Manufactured housing 3 – Multifamily
Property Type	1 – One to four-family (other than manufactured housing) 2 – Manufactured housing 3 – Multifamily
Loan Purpose	1 – Home purchase 2 – Home improvement 3 – Refinancing
Owner_occupancy	1 – Owner-occupied as a principal dwelling 2 – Not owner-occupied 3 – Not applicable
Loan Amount	in thousands of dollars
Preapproval	1 – Pre Approval was requested 2 – Pre Approval was not requested 3 – Not applicable
Action_taken	1 – Loan originated 2 – Application approved but not accepted 3 – Application denied by financial institution
Applicant_ethnicity	1 – Hispanic or Latino 2 – Not Hispanic or Latino 3 – Information not provided by applicant in mail, Internet, or telephone application 4 – Not applicable 5 – No co-applicant
Co-applicant Ethnicity	Same as above
Applicant Race	1 – American Indian or Alaska Native 2 – Asian 3 – Black or African American 4 – Native Hawaiian or Other Pacific Islander 5 – White 6 – Information not provided by applicant in mail, Internet, or telephone application 7 – Not applicable

Continued on next page

Table 1 – continued from previous page

Variables	Descriptions
	8 – No co-applicant
Applicant Sex	1 – Male 2 – Female 3 – Information not provided by applicant in mail, Internet, or telephone application 4 – Not applicable 5 – No co-applicant
Applicant Income	in thousands of dollars
Hud median family income	FFIEC Median family income in dollars for the MSA/MD in which the tract is located (adjusted annually by FFIEC)
Tract to msamd income	% of tract median family income compared to MSA/MD median family income. (Carried to two decimal places)
Number of owner occupied units	Number of dwellings, including individual condominiums that are lived in by the owner
Number of 1 to 4 family units	Dwellings that are built to house fewer than 5 families
Population	Total population in tract
Minority_population	Percentage of minority population to total population for tract

2.2. *EDA Parts*

We choose "action taken" as our response variable. The action_taken equals 1 means the loan got approved. According to Figure 15a, through adding up the first two types of actions, nearly 79% of loan applications got approved.

To have a deeper understanding of the relationships between our chosen predictors and the response variable, we generate multiple plots, which also shows the distributions of action taken in different values of independent variables.

Figure 15b and Figure 15c show the visualization of the loan. 'Home purchase', 'Home improvement' and 'Refinancing' are indexed as 1, 2, and 3 respectively. Among the three existing loan purposes, 'Home purchase' is the most common one as over half of the mortgage applicants applied for loans for this reason. Meanwhile, it is interesting to see that the loan purpose called 'home improvement' is least likely to get approved, as more applicants got rejected than accepted under this loan purpose. Besides, the density plot shows the loan amount distribution in 2017. The loan_amount value follows the normal distribution and the median loan amount seems to be around $200,000.

Figure 15d shows the distribution of the six most frequent agency codes. It is quite clear that agencies called the Department of Housing and Urban Development(HUD) and Consumer Financial Protection Bureau (CFPB) got the most applicants. Comparatively, National Credit Union Administration (NCUA) has the highest approval rate among the six agencies.

The following four graphs show the distribution of different features of applicants' information. The loan approval and denial rate is falling under each category of the following four predictors. Figure 15e, Figure 15f, and Figure 15g show detailed information related to the applicant's race, ethnicity, and sex. The number of male applicants is about 2 times higher than the number of female applicants. Figure 15h shows the density plot of applicant income. The distribution of applicant income also seems to be normally distributed, meanwhile, the median of applicant's income is around \$100,000 according to the density plot.

3. Data-preprocessing

We preprocess our data from scratch, including handling NA's and outliers, encoding the categorical variables, standardization, dealing with unbalanced data, and principal component analysis.

Also, we divide the dataset into training and testing sets according to the ratio of 70-30. It is worth pointing out that due to a large amount of data and extremely slow speed of R, we sample 1,000 records out of all data for training and testing as a demo.

3.1. *Cleaning and encoding*

3.1.1. *NA's*

Many columns are mainly consisting of NA's, which provides almost no information, so we drop such variables. After that, we drop the rows that contain NA's, because the data volumes are so big, and we believe removing such rows would not change the distribution of our data. Detailed information of missing data can be viewed in Figure 9: Percentage of missing data by features.

3.1.2. *Outliers*

Some variables contain some extreme values, like the variables regarding incomes. They may affect the model training and even some of them are wrong data. We remove rows that are corresponding to the top 4% quantile values. The threshold is arbitrary and the purpose is to remove some wrong records while keeping the distribution of the variables.

3.1.3. *Standardization*

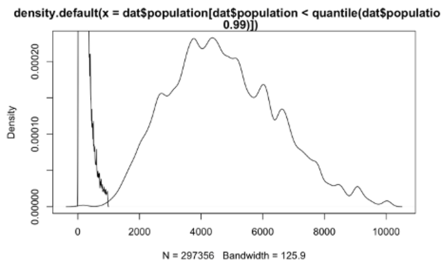
Some models' computations are based on distance, so if one of the features has a broad range of values, the distance will be affected by it and make this particular feature too important in the model. We use a z-score to standardize the numeric variables, which is like:

$$x_{stand} = \frac{x - mean(x)}{sd(x)}$$

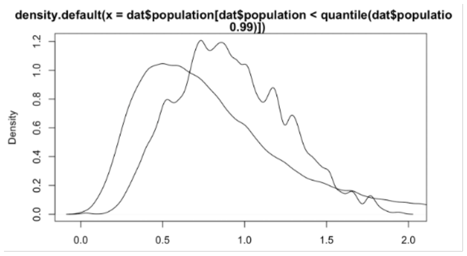
From Figure 1 , we can see that the data are scaled into the same range.

3.1.4. *Encoding/One-hot encoding*

Categorical variables can be represented by words, but numbers are better. Different classes are encoded by different numbers, and what's more important is their ‘class()‘ in R. They should be ‘factor()‘ and never used as numbers in any models. Also, we use one-hot encoding to generate dummy variables corresponding to different classes. If the specific function in R can handle variables of class ‘factor()‘, that is good. However, the one-hot encoding is a much more general method.

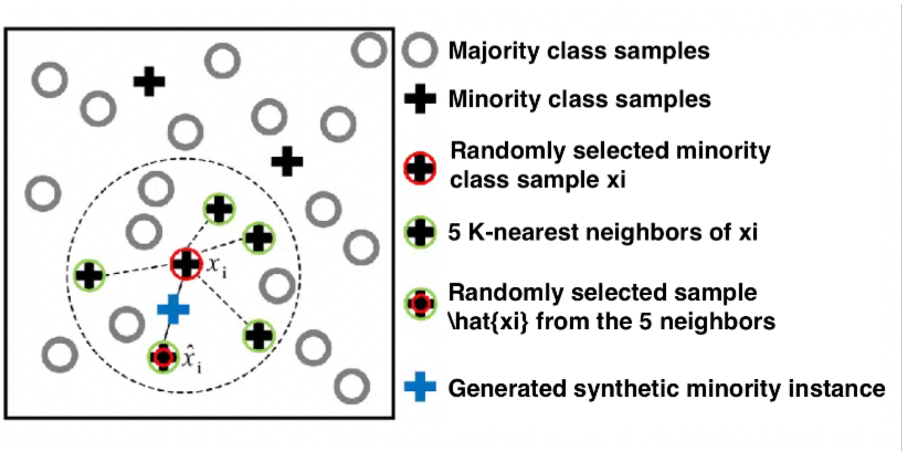


(a) Before Standardization



(b) After Standardization

Figure 1: Standardization



Majority class samples
Minority class samples
Randomly selected minority class sample xi
5 K-nearest neighbors of xi
Randomly selected sample \hat{xi} from the 5 neighbors
Generated synthetic minority instance
x_i
$\hat{x}_i$

Source: https://rikunert.com/smote_explained

Figure 2: SMOTE Algorithm

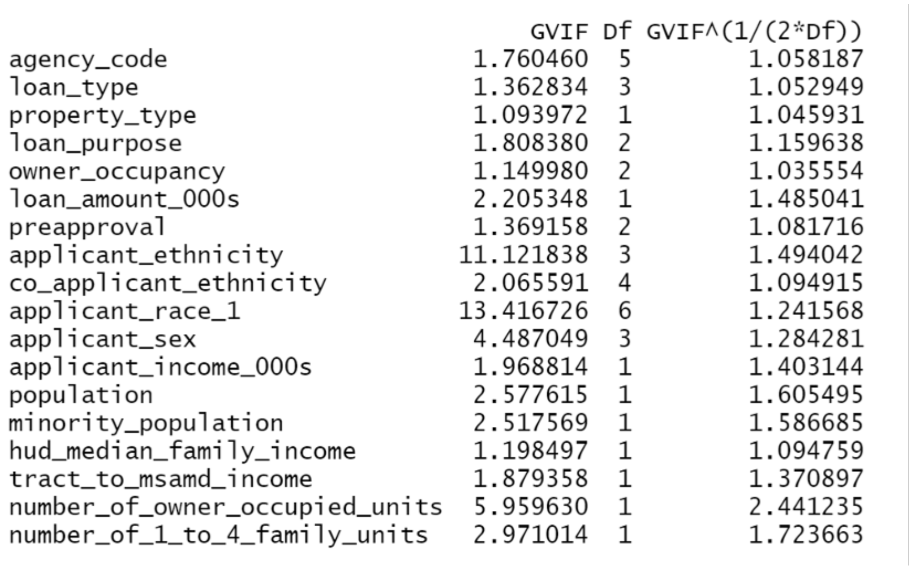
3.2. *Unbalanced data: SmoteNC*

From the descriptive part above, we know that the data is unbalanced. There are many more people who get accepted than those who are rejected in loan requests. The unbalancedness may make the model not capture enough features of the minority class. Also, it would strongly influence the evaluation of the model prediction results. We use Synthetic Minority Over-sampling TEchnique-Nominal Continuous (SMOTE-NC) to handle this problem. It was created by Chawla, et.al. (2002). The algorithm is based on the Euclidean distance between the feature vectors for which k-nearest neighbors are being identified (minority class sample). The Figure 2 shows its basic concepts.

In our experiment, after SMOTE-NC, the labels of the dataset become balanced (536:536, previous 146:536).

3.3. *Principal components for mixed data: FAMD*

Before creating the model, we check for multicollinearity. If there are high correlations between our variables, we need to deal with that problem. Variance Inflation Factor is a measure of multicollinearity for the variables, but it is usually applied to non-categorical variables. Since our model includes a mixture of categorical and non-categorical vari-



```
                                     GVIF Df GVIF^(1/(2*Df))
agency_code                      1.760460  5        1.058187
loan_type                        1.362834  3        1.052949
property_type                    1.093972  1        1.045931
loan_purpose                     1.808380  2        1.159638
owner_occupancy                  1.149980  2        1.035554
loan_amount_000s                 2.205348  1        1.485041
preapproval                      1.369158  2        1.081716
applicant_ethnicity             11.121838  3        1.494042
co_applicant_ethnicity           2.065591  4        1.094915
applicant_race_1                13.416726  6        1.241568
applicant_sex                    4.487049  3        1.284281
applicant_income_000s            1.968814  1        1.403144
population                       2.577615  1        1.605495
minority_population              2.517569  1        1.586685
hud_median_family_income         1.198497  1        1.094759
tract_to_msamd_income            1.879358  1        1.370897
number_of_owner_occupied_units   5.959630  1        2.441235
number_of_1_to_4_family_units    2.971014  1        1.723663
```

Figure 3: GVIF

ables, we calculate the Generalized Variance Inflation Factor (GVIF) instead. GVIF is calculated as

$$GVIF = VIF^{\frac{1}{2*df}}$$

Our variables' GVIF results are shwon in Figure 3.

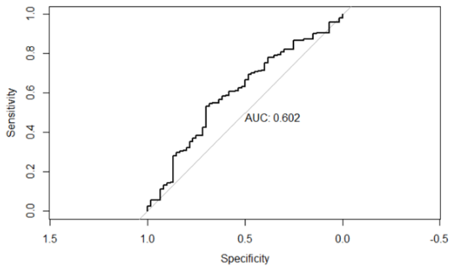
To check if there is a sign of multicollinearity, we compare the GVIF values with $10^{\frac{1}{2*df}}$. Thus, we find that applicant_ethnicity and applicant_race_1 show a sign of multicollinearity.

In order to solve the multicollinearity problem and to filter out the variables with less important information, we consider the Principle Component Analysis method. We want to use the PCA algorithm to reduce dimensionality and extract the most important uncorrelated variables.

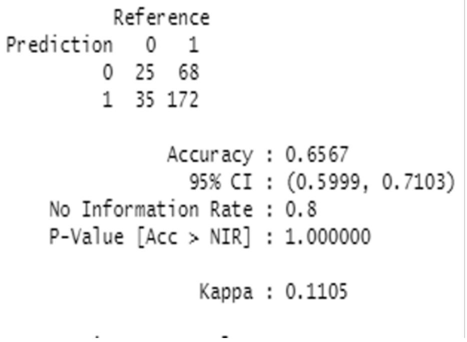
However, since our data includes both quantitative and qualitative variables, the normal PCA method is not applicable. We have to either use one-hot encoding to transform the categorical variables or use the Factorial Analysis of Mixed Data (FAMD) method. In fact, the one-hot encoding method does not work well. When we transform the categorical variables, one categorical variable will be changed to several variables depending on the number of the categories. Then, the results of PCA will not assign similar weights as the original variables. Therefore, we choose FAMD as a preprocessed method. FAMD applies the PCA algorithm on both quantitative and qualitative variables and assigns the same weights to variables.

We apply the "FAMD" function in R directly on our training data, and store the coordinates as our new estimates. The new model has eight dimensions, which can be used as preprocessed data for logistic regression, Random Forest model, and XGBoost model. The preprocessed data improves the logistic model, but not for Random Forest or XGBoost.

One problem for the FAMD preprocessed data is the loss of interpretation. Though the PCA method performs well in dealing with multicollinearity, it only solves the problem



(a) ROC Curve



```
           Reference
Prediction   0   1
         0  25  68
         1  35 172

               Accuracy : 0.6567
                 95% CI : (0.5999, 0.7103)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1.000000

                  Kappa : 0.1105
```

(b) Confusion Matrix

Figure 4: Results for Logistic Regression

numerically. PCA rotates the data and creates new dimensions and our eighteen variables are shrunk to eight. Hence, it is difficult to interpret the variables for the new model.

Another problem is that the decision tree is immune to multicollinearity. Since Random Forest and XGBoost are based on decision trees, they are not affected by multicollinearity. Thus, using the FAMD method does not improve our model. Details will be shown in the Random Forest and XGBoost model section.

4. Models

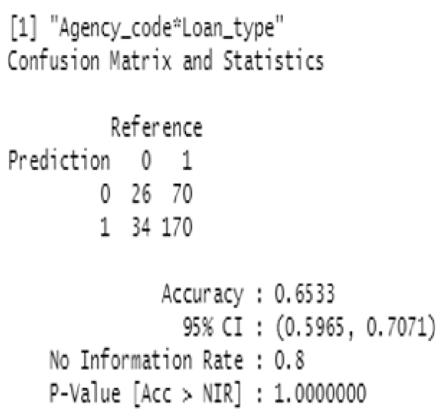
4.1. *Logistic regression*

We first try to use logistic regression to fit the model. Logistic regression is a kind of predictive analysis that explains the relationship between one dependent variable and some independent variables. The reason why we choose logistic regression is that logistic regression can be used when the dependent variable (action_taken) is binary and it can work out the probability of success of an event. To be specific, if the probability greater than 0.5 is classified as loan originated, otherwise, it is classified as loan denial. Moreover, logistic regression supports categorical data and about 55% of our independent variables such as loan type and loan purpose can be converted to categorical variables. Finally, logistic regression is easy to implement and interpret. Therefore, it might be possible to work out an accurate model directly from logistic regression.

From Figure 4, we can see that the accuracy of the model is 65.67%. However, the percentage of people who get a loan in the testing set is 80%, which means the logistic regression model is much worse than a model that simply says everyone can get a loan.

There are some reasons why logistic regression does not perform as well as we expected. The first reason might be the size of the dataset we used. A dataset of size 700 may not be large enough to include all cases of 19 variables. The second problem is that logistic regression assumes that the dependent variables and the independent variables have a linear relationship. However, variables such as applicant income and loan amount may not have a linear relationship with the probability of getting loans in reality. The third problem is that we might have ignored some interactions that affect the accuracy of prediction.

We first try to investigate the interaction effect, and we are interested about interactions between agency name, loan type and loan purpose, as we think there must be some link between these three variables.

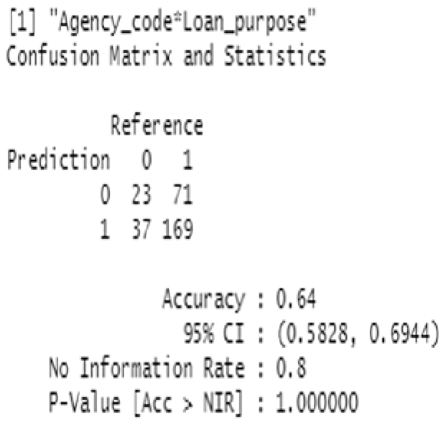


```
[1] "Agency_code*Loan_type"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  26  70
         1  34 170

               Accuracy : 0.6533
                 95% CI : (0.5965, 0.7071)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1.0000000
```

(a) Interaction 1

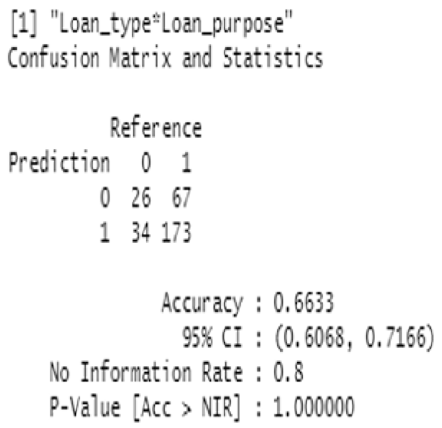


```
[1] "Agency_code*Loan_purpose"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  23  71
         1  37 169

               Accuracy : 0.64
                 95% CI : (0.5828, 0.6944)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1.000000
```

(b) Interaction 2

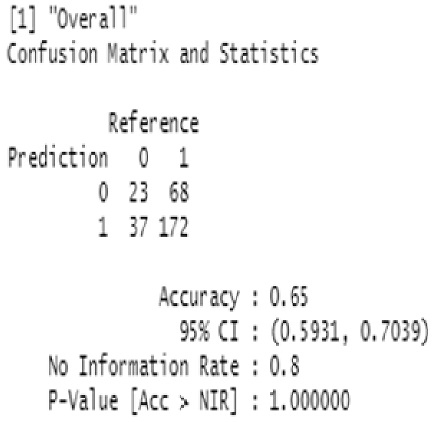


```
[1] "Loan_type*Loan_purpose"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  26  67
         1  34 173

               Accuracy : 0.6633
                 95% CI : (0.6068, 0.7166)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1.000000
```

(c) Interaction 3



```
[1] "Overall"
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  23  68
         1  37 172

               Accuracy : 0.65
                 95% CI : (0.5931, 0.7039)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1.000000
```

(d) Interaction 4

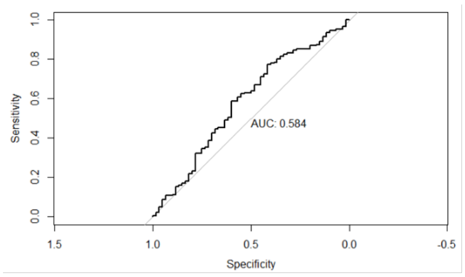
Figure 5: Tests for Interactions

For example, a person who wants to buy a house may have a higher chance of getting a loan when he applies for a loan insured by the Federal Housing Administration, and this may help us improve the accuracy of prediction.

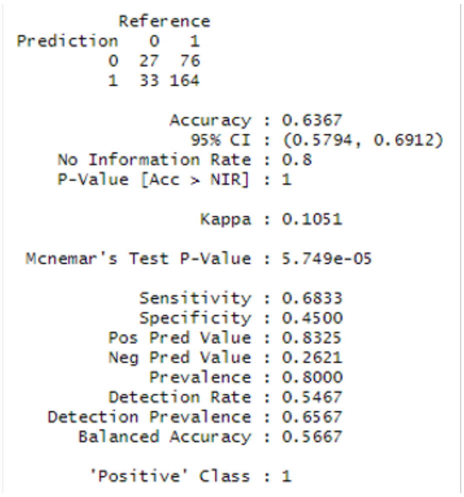
And the way we evaluate the effectiveness of improvement of a single interaction is by calculating the accuracy of the model with that interaction and then comparing it to the accuracy of the model without any interactions.

As shown in Figure 5, adding interactions between any pairs of variables does not improve the accuracy of the model effectively. And when we added all three interactions into the model, the accuracy of the model is still 65%. Therefore, we consider that adding interaction effects between variables does not significantly improve our model.

Finally, we fit the logistic regression model to our FAMD processed dataset. The result is shown in Figure 6. The accuracy of the model is worsened from 65.67% to 63.67%, and it suggests that multicollinearity might not be a problem in our original dataset.



(a) ROC Curve



```
          Reference
Prediction   0   1
         0  27  76
         1  33 164

               Accuracy : 0.6367
                 95% CI : (0.5794, 0.6912)
    No Information Rate : 0.8
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1051

 Mcnemar's Test P-Value : 5.749e-05

            Sensitivity : 0.6833
            Specificity : 0.4500
         Pos Pred Value : 0.8325
         Neg Pred Value : 0.2621
             Prevalence : 0.8000
         Detection Rate : 0.5467
   Detection Prevalence : 0.6567
      Balanced Accuracy : 0.5667

       'Positive' Class : 1
```

(b) Confusion Matrix

Figure 6: Results for Logistic regression Combined with FAMD

4.2. *Artificial Neural network*

Among different types of neural networks, we choose the classic multilayer perceptron (MLP), which is based on a supervised procedure and consists of three layers: input, hidden, and output. It can handle both numeric variables and categorical variables by one-hot encoding. Also, we choose the resilient backpropagation learning algorithm, which is better than normal backpropagation.

Due to the extremely slow speed in R, we are not able to try a lot of different network structures. Figure 7 below shows the structure we tried.

Results are shown in Figure 8, we can see that the model still can not capture the features why people get rejected and predict more True Negatives to be positive. As for the True positive samples, it performs well.

Hyperparameters selection for neural networks and enough training time is of vital importance to a neural network. More complex or deep structures of neural networks should be tried systematically. After better hyperparameters selection, the performance of neural networks may improve.

4.3. *Random forest*

The next attempt is using Random forest. Random forest is a type of machine learning technique that can be used for classification problems. It has some main advantages when working on our dataset. Firstly, it can work for both categorical and continuous variables. In our dataset, we have categorical variables like loan type and loan purpose, and we have continuous variables like applicant income and loan amount. Moreover, it works well with non-linear parameters. If applicant income does not have a linear relationship with the probability of getting a loan, the random forest will be a good choice dealing with this problem. Therefore, Random forest may be a better fit compared to logistic regression.

Then we start to fit a random forest. The function is shown in Figure 9. One important parameter in the random forest is mtry, it is the number of variables randomly sampled as candidates at each split. We have 19 variables, so we can have mtry ranging from 1 to 19. And the accuracy of each fit is shown in Figure 10.

We can see that mtry=2 is the optimal choice as the accuracy of prediction is the

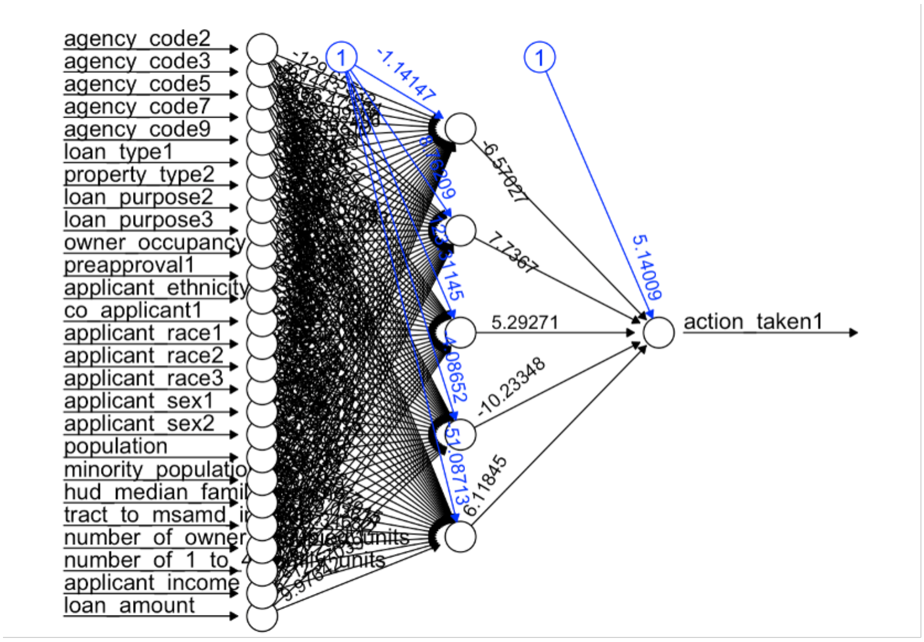


Figure 7: Structure of Our ANN Model

highest, and the accuracy of prediction is 71.00%. It is more accurate compared to the logistic model. Although it is still worse than a model that simply says everyone can get a loan, we still consider this to be a good result as there is a huge improvement compared to the previous models.

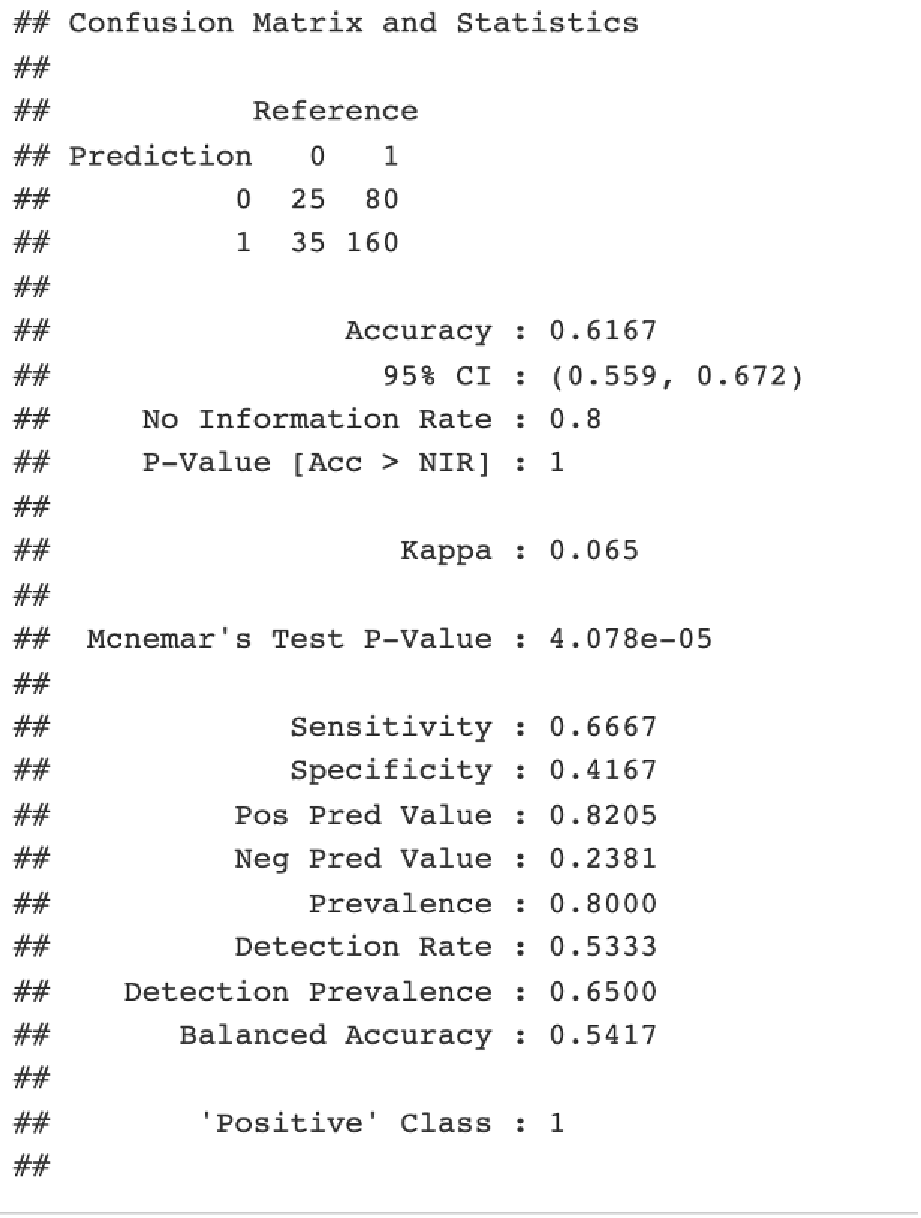
4.4. *XGBoost*

After trying the tree model, we do not limit our scope of exploration as we try a new method called XGBoost, using decision trees as base learners. We hold the faith that this method might yield better results, as it is combining many weak learners to make a strong learner and is well-known as an ideal and efficient implementation of gradient boosting that can be used for classification predictive modeling. Meanwhile, XGBoost is a great fit for our dataset as it is well suited especially those related to real-world business problems such as fraud detection or customer churn prediction.

When we fit the XGBoost model, we use both training datasets that implemented SMOTE and FAMD balanced methods respectively. Randomness is used in the construction of the model. We also run cross-validation after each iteration. The result of the AUC and accuracy rate we get here is 0.612 and 0.7234 for the model using the SMOTE training datasets. Meanwhile, the AUC and accuracy rate we get for the FAMD balanced training dataset is 0.591 and 0.6632. It is obvious that the XGBoost model using the SMOTE regularization training dataset performs better than the XGBoost model using the FAMD balanced regularization. The XGBoost model using the SMOTE regularization also has a slightly higher AUC and accuracy rate than the random forest method.

Figure 11 selects significant predictors of the SMOTE regularized XGBoost model.

Although the XGBoost method is highly flexible, it still contains many shortcomings which may explain the unsatisfying result we got here. XGBoost does not perform well on sparse and unstructured data. Although we have already tried several methods to

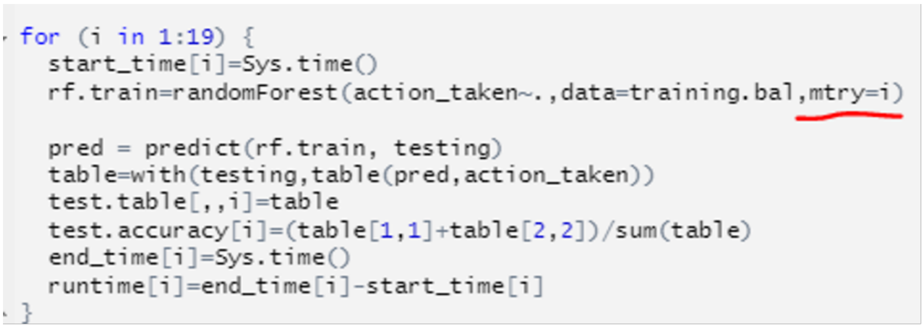


```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  25  80
##          1  35 160
##
##                Accuracy : 0.6167
##                  95% CI : (0.559, 0.672)
##     No Information Rate : 0.8
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.065
##
##  Mcnemar's Test P-Value : 4.078e-05
##
##             Sensitivity : 0.6667
##             Specificity : 0.4167
##          Pos Pred Value : 0.8205
##          Neg Pred Value : 0.2381
##              Prevalence : 0.8000
##          Detection Rate : 0.5333
##    Detection Prevalence : 0.6500
##       Balanced Accuracy : 0.5417
##
##        'Positive' Class : 1
##
```

Figure 8: Structure of Our ANN Model

solve the highly sparse feature of our datasets, it may still cause much trouble when we try to apply the XGBoost method.

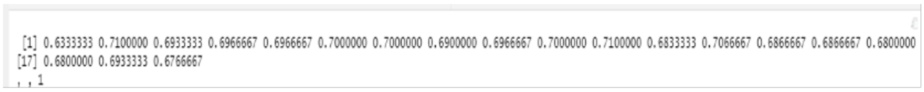
For further expansion and improvement on this method, it is well-worth for us to keep trying and repeating the modeling process or performing systematic experimentation such as using a grid search across a range of values to find better hyperparameters. Meanwhile, when fitting a final model, it may be desirable to either increase the number of trees until the variance of the model is reduced across repeated evaluations, or to fit multiple final models and average their predictions.



```
for (i in 1:19) {
  start_time[i]=Sys.time()
  rf.train=randomForest(action_taken~.,data=training.bal,mtry=i)

  pred = predict(rf.train, testing)
  table=with(testing,table(pred,action_taken))
  test.table[,,i]=table
  test.accuracy[i]=(table[1,1]+table[2,2])/sum(table)
  end_time[i]=Sys.time()
  runtime[i]=end_time[i]-start_time[i]
}
```

Figure 9: A Function to Test Different Hyperparameters for Random Forest



```
 [1] 0.6333333 0.7100000 0.6933333 0.6966667 0.6966667 0.7000000 0.7000000 0.6900000 0.6966667 0.7000000 0.7100000 0.6833333 0.7066667 0.6866667 0.6866667 0.6800000
[17] 0.6800000 0.6933333 0.6766667
, , 1
```

Figure 10: Accuracy of Different mtry of Random Forest

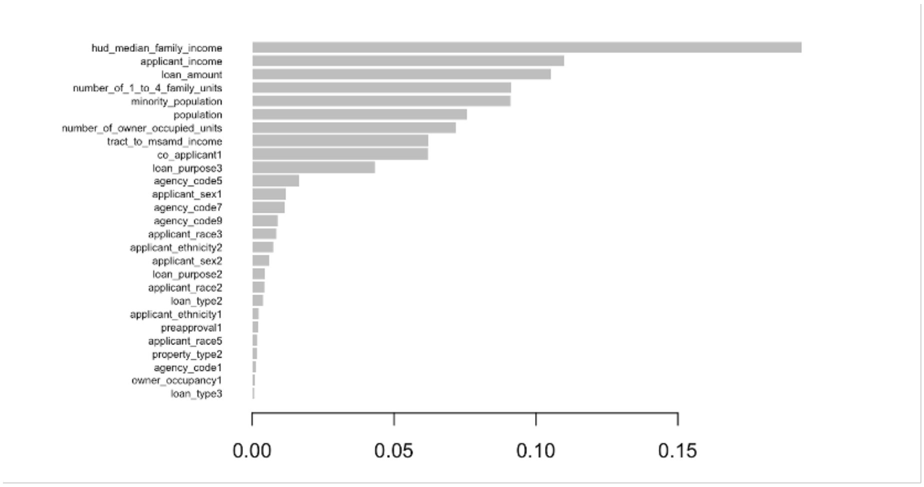


Figure 11: Importance of Variables in XGBoost

4.5. *SVM*

From the previous data description section, we can see that all the independent variables in the data are labeled, which is loan denial or loan originated. This suggests that we can try to use SVM, which is a supervised learning model to classify two-type observations. There are potential benefits to using SVM, for example, it is very useful when data has a number of independent variables greater than the data size. Also, it can work efficiently in high-dimensional space.

For SVM, We first find the points that lie closest to both the classes, which are support vectors. Then we try to find the proximity between our dividing plane and the support vectors. The distance between the points and the dividing line is called margin. What SVM is trying to do is to maximize the margin. When the margin is maximized, the hyperplane is what we are looking for.

For the simple case, if we only have two features, the hyperplane will be 1 dimension,

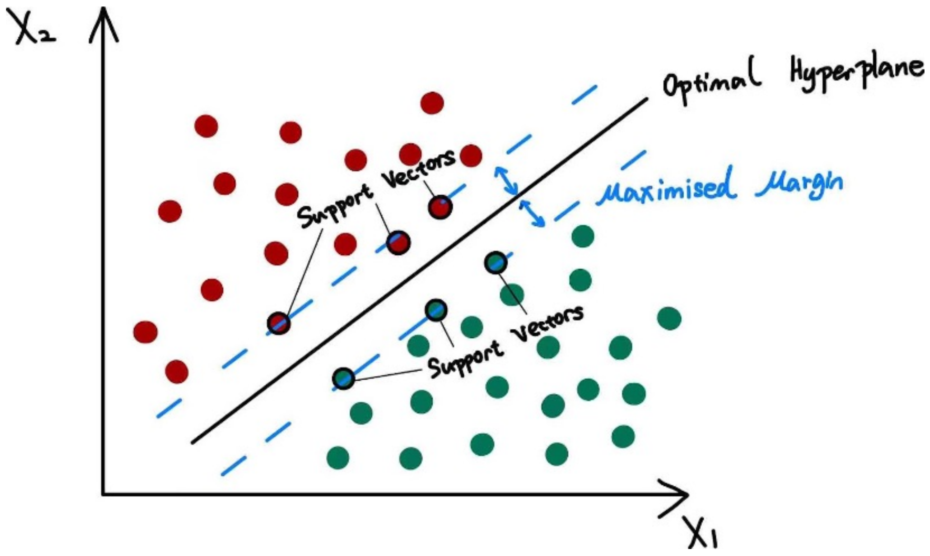
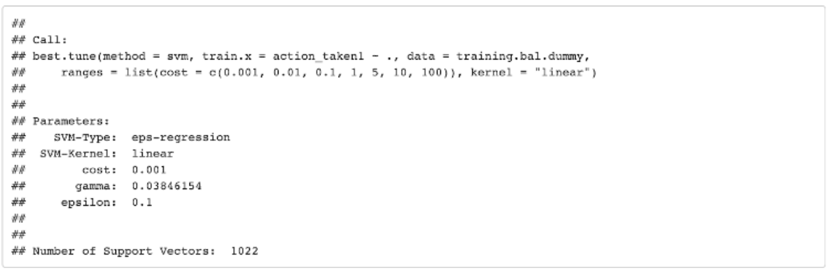


Figure 12: SVM Algorithm



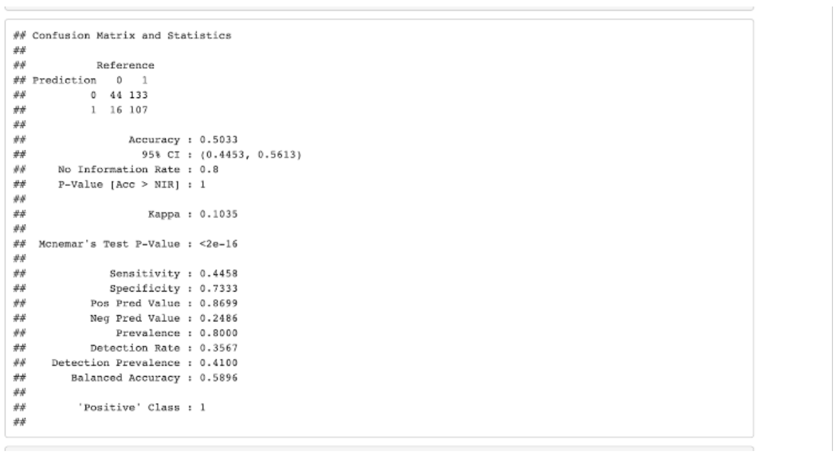
```
##
## Call:
## best.tune(method = svm, train.x = action_taken1 ~ ., data = training.bal.dummy,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  0.001
##       gamma:  0.03846154
##     epsilon:  0.1
##
##
## Number of Support Vectors:  1022
```

Figure 13: SVM Cost Selection

which is a line. (in Figure 12) Because for this project, we have 10 features, we have a 9-dimensional subset that divides the space into two disconnected components.

What we do in this project is to use a linear classifier by using a package "e1071" which contains the SVM function in R. Next we need to decide the cost argument in the SVM function because the cost quantifies the penalty to an observation on the wrong side of the classification boundary(misclassification). So we use the tune() function to test a sequence of costs and identify the value that gives the best fitting model. In this case, the cost = 0.1.(see Figure 13)

Then, we try to come up with a model by setting cost = 0.1 in the SVM function. Then we use that model to predict the testing data, which is 30% of our whole data. The SVM calculates the probability that a person gets approved, and we classify the probability greater than 0.5 as approval, otherwise, it is rejected. Then we get the confusion matrix, which is shown in Figure 14. The error case is that the loan originated but the model classifies it as denial (False Positive) is about 8 times more than the case where the model misclassified the loan as originated but the loan is actually denied (False Negative). But we do not know the exact reason for this, which needs more future research. The accuracy of the model produced by SVM does not predict very well as the random forest model



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  44 133
##          1  16 107
##
##                Accuracy : 0.5033
##                  95% CI : (0.4453, 0.5613)
##     No Information Rate : 0.8
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1035
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.4458
##             Specificity : 0.7333
##          Pos Pred Value : 0.8699
##          Neg Pred Value : 0.2486
##              Prevalence : 0.8000
##          Detection Rate : 0.3567
##    Detection Prevalence : 0.4100
##       Balanced Accuracy : 0.5896
##
##        'Positive' Class : 1
##
```

Figure 14: Results for SVM

because the accuracy is only 50.33%. Even though the data we are using here is balanced data, it does not help to improve this model.

Besides, SVM has some disadvantages. When the data size is too large, the model is not expected to work well for a long training time. This is what we came across when we were doing this model, we first tried the 5,000 data for the first time. But the SVM was too slow to generate the result. Then, we changed to trying 1,000 data, which SVM becomes much faster. This is the reason why the testing data is only 300.

Besides, SVM has some disadvantages. When the data size is too large, the model is not expected to work well for a long training time. This is what we came across when we were doing this model, we first tried the 5000 data for the first time. But the SVM is too slow to generate the result. Then, we changed to trying 1000 data, which SVM becomes much faster. This is the reason why the testing data is only 300.

5. Conclusions

All in all, the models we tried before do not perform very well. We have tried a lot of different measures to improve the model like rebalancing the data, but none of them shows significant improvements. However, from all our models, we can see there are some variables that always contribute a lot, like applicant income, loan amount, and loan purpose. It is reasonable to do the following research into loan origination.

6. Problem and Limitations

During the modeling of our dataset, we realize that there is another severe problem that holds us back from creating an accurate model, which is the lack of some key variables. For example, the dataset does not contain any information about peoples' credit record, while credit record is one of the main factors that banks consider. Apart from credit records, variables such as employment status and bank savings are also missing from our dataset. The lack of key variables makes it hard to analyze banks' behavior because they may behave differently while facing people of identical information. This would lead to misclassification while modeling the data and reduce the accuracy of our model. Those omitted variables in our data set will be fully discussed in the following section. We will dig into the problem in the next section.

7. Omitted variables

Some regular criterias for applying for a mortgage successfully include a steady income and a dependable working history. It will be difficult for unemployed people to apply for a mortgage. A co-borrower can be a strategy to get a mortgage after unemployment since the loan lender will consider the co-borrower's income and his/her qualifications for the mortgage. Hence, a person who is employed or not may not be a big part of determining whether he/she can get the mortgage or not.

Another variable might be the Loan-to-value ratio. According to the Consumer Financial Protection Bureau, the Loan to Value ratio needs to be taken into consideration as determining whether to lend the borrowers money or not. LTV is calculated as the amount of the loan compared to the appraised value of the property (LTV Ratio). Higher LTV ratios for borrowers show higher risks to lenders. For our data, we have a "loan_amount" variable but need the "value of the property" to calculate the LTV ratio. The Department of the Treasury also emphasizes the importance of the appraised value of the mortgage, "While borrowers' ability to repay their real estate loans according to reasonable terms remains the primary consideration in the lending decision, an institution also must consider the value of the underlying real estate collateral in accordance with the Agencies' appraisal regulations" (Department of the Treasury, page 8). It issued Interagency Appraisal and Evaluation Guidelines, providing appraisal standards for evaluations and examiners. Some basic examination contents include the location of the property, current and projected use of the property, estimated market value in the actual physical condition, and property-specific data (Department of the Treasury, page 13). However, these are not the only factors that influence the appraised value. The evaluation methods vary depending on different agencies and the validity may change due to the time and market conditions. Therefore, we need more data to calculate the LTV ratio to improve our model.

In addition, an article on the Time points out several standards that mortgage underwriting is looking at like credit history, Debt-to-income ratio, income verification, and so on (Lauren). For our project, the data set does not have information about the credit history of an individual. As mentioned in the article, those are very important to determine if a person's mortgage loan gets approved.

Other essential factors are included in the FICO score. Wells Fargo provides the constitution of FICO scores, ranging from 300 to 850. The higher the FICO score, the less risky the borrowers (Wells Fargo). Table 2 shows factors that affect FICO scores. Due to the lack of data, the model might generate some errors when predicting.

Table 2: Factors for FICO Score

FICO Score Factors	**Descriptions**
Payment History (35%)	Payments on time Frequency of missed payments Passed due dates Recent missed payments Payments 30 days past due Number of accounts (showing late payment)
Money Owed (30%)	Entire amount owed Number and types of accounts Proportion of money owed compared to credit available

Continued on next page

Table 2 – continued from previous page

FICO Score Factors	**Descriptions**
Length of credit history (15%)	Average age of credit
Types of accounts (10%)	Mix of accounts Installment loans Home loans Retail and credit cards
Recent credit activity (10%)	Recently opened accounts Other high risky actions

8. References

Berry, Craig. "Can You Buy a House or Get a Mortgage on Unemployment?" Mortgage Rates, Mortgage News and Strategy: The Mortgage Reports, 6 Aug. 2021, https://themortgagereports.com/70823/buy-a-house-on-unemployment-income.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Department of the Treasury. "Interagency Appraisal and Evaluation Guidelines". Journal Register, vol. 75, no. 237, 10 Dec. 2010, https://www.govinfo.gov/content/pkg/FR-2010-12-10/pdf/2010-30913.pdf. Accessed 5 Dec. 2021.

LTV Ratio, "What Is a Loan-to-Value Ratio and How Does It Relate to My Costs?", Consumer Financial Protection Bureau, 9 Sept. 2020, https://www.consumerfinance.gov/ask-cfpb/what-is-a-loan-to-value-ratio-and-how-does-it-relate-to-my-costs-en-121.

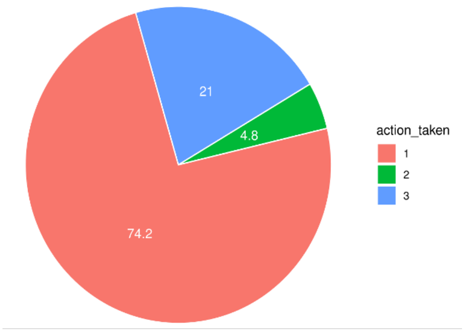
Pupale, Rushikesh. "Support Vector Machines(Svm) - an Overview." Towards Data Science, 11 Feb. 2019, https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989.

Thangavelu, Poonkulali. "How the Federal Reserve Affects Mortgage Rates?" Investopedia, Investopedia, 3 Dec. 2021, https://www.investopedia.com/articles/personal-finance/050715/how-federal-reserve-affects-mortgage-rates.asp.

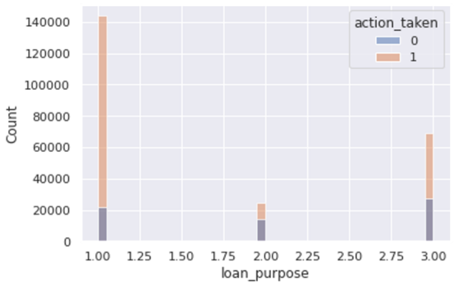
Wells Fargo, "How Your Credit Score Is Calculated." Wells Fargo, https://www.wellsfargo.com/financial-education/credit-management/calculate-credit-score/.

Ward, Lauren. "What Is Mortgage Underwriting? Explaining The Underwriting Process." Time, Time, 24 Aug. 2021, https://time.com/nextadvisor/mortgages/underwriting-process.

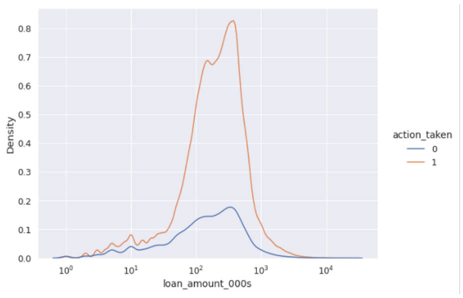
Appendix A. Exploratory Data Analysis (EDA)



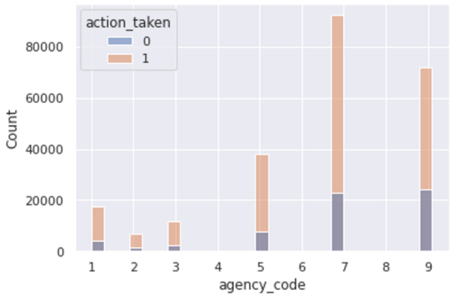
(a) Visualization of dependent variable



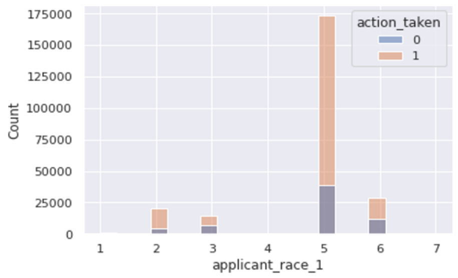
(b) Visualization of loan



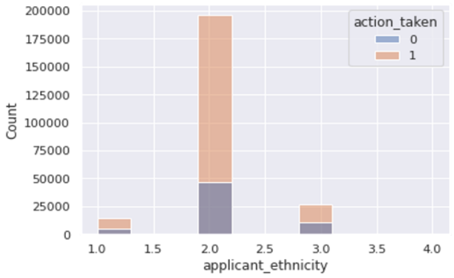
(c) Density plot of loan



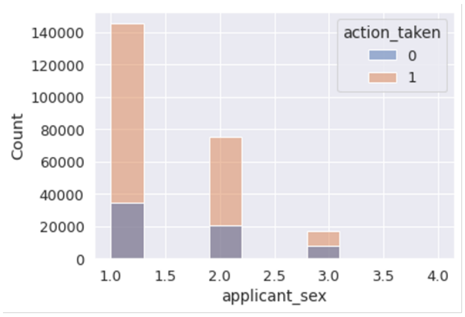
(d) Visualization of agency code



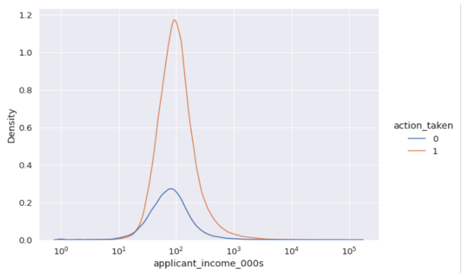
(e) Visualization of applicant race



(f) Visualization of applicant ethinicity



(g) Visualization of applicant sex



(h) Density plot of applicant income

Figure 15: Exploratory Data Analysis (EDA): Part 1

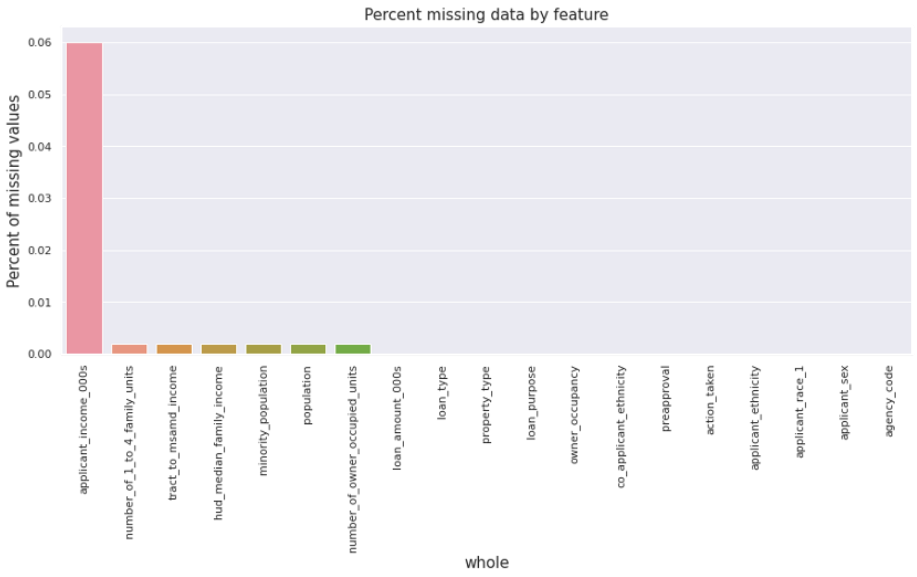


Figure 16: Exploratory Data Analysis (EDA): Part 2: Percentage of missing data by features

Appendix B. R Code

```
library(tidyverse)
library(glmnet)
library(caret)
library(pROC)

dat <- read.csv("/Users/yctang/Documents/Columbia/5291␣Advanced␣data␣
    ↪ Analysis/project/data/hmda_2017_ny_all-records_labels.csv")
# remove some variables & NA
dat <- subset(dat, select = -c(1:4, 6, 8, 10, 12, 15, 17, 19:27, 29, 31,
    ↪ 33:50, 51, 53, 54, 56:71, 69:71, 78))

## Data Pre-processing
# drop missing values
dat <- drop_na(dat)

# some adjustments: re-encoding
# owner_occupancy: other == 0, Owner-occupied as a principal dwelling ==
    ↪ 1
dat[dat$owner_occupancy == 2, ]$owner_occupancy <- 0
dat[dat$owner_occupancy == 3, ]$owner_occupancy <- 0
# loan_type: other == 0, Conventional == 1
dat[dat$loan_type == 2, ]$loan_type <- 0
dat[dat$loan_type == 3, ]$loan_type <- 0
dat[dat$loan_type == 4, ]$loan_type <- 0
# preapproval: not requested == 0
dat[dat$preapproval == 2, ]$preapproval <- 0
dat[dat$preapproval == 3, ]$preapproval <- 0
# action_taken
dat = dat %>% filter(action_taken == 1 | action_taken == 2 | action_taken
    ↪  == 3)
dat[dat$action_taken == 2, ]$action_taken <- 1
dat[dat$action_taken == 3, ]$action_taken <- 0
# applicant_ethnicity: other == 0, hispanic/latino == 1
dat[dat$applicant_ethnicity == 2, ]$applicant_ethnicity <- 0
dat[dat$applicant_ethnicity == 3, ]$applicant_ethnicity <- 0
dat[dat$applicant_ethnicity == 4, ]$applicant_ethnicity <- 0
# sex: unknown == 0, male == 1, female == 2
dat[dat$applicant_sex == 3, ]$applicant_sex <- 0
dat[dat$applicant_sex == 4, ]$applicant_sex <- 0
# race: other == 0, asian == 1, Black or African American == 2, white ==
    ↪ 3
dat[dat$applicant_race_1 == 1, ]$applicant_race_1 <- 0
dat[dat$applicant_race_1 == 4, ]$applicant_race_1 <- 0
dat[dat$applicant_race_1 == 6, ]$applicant_race_1 <- 0
dat[dat$applicant_race_1 == 7, ]$applicant_race_1 <- 0
dat[dat$applicant_race_1 == 2, ]$applicant_race_1 <- 1
dat[dat$applicant_race_1 == 3, ]$applicant_race_1 <- 2
dat[dat$applicant_race_1 == 5, ]$applicant_race_1 <- 3
```

```
names(dat)[names(dat) == "applicant_race_1"] <- "applicant_race"
# co-applicant, yes/no == 1/0
dat[dat$co_applicant_ethnicity == 2, ]$co_applicant_ethnicity <- 1
dat[dat$co_applicant_ethnicity == 3, ]$co_applicant_ethnicity <- 0
dat[dat$co_applicant_ethnicity == 4, ]$co_applicant_ethnicity <- 0
dat[dat$co_applicant_ethnicity == 5, ]$co_applicant_ethnicity <- 0
names(dat)[names(dat) == "co_applicant_ethnicity"] <- "co_applicant"

# Change classes of variables
dat$agency_code <- as.factor(dat$agency_code)
dat$loan_type <- as.factor(dat$loan_type)
dat$property_type <- as.factor(dat$property_type)
dat$loan_purpose <- as.factor(dat$loan_purpose)
dat$owner_occupancy <- as.factor(dat$owner_occupancy)
dat$preapproval <- as.factor(dat$preapproval)
dat$action_taken <- as.factor(dat$action_taken)
dat$applicant_ethnicity <- as.factor(dat$applicant_ethnicity)
dat$co_applicant <- as.factor(dat$co_applicant)
dat$applicant_race <- as.factor(dat$applicant_race)
dat$applicant_sex <- as.factor(dat$applicant_sex)

## Data Visualization

dat %>%
  ggplot(aes(action_taken, fill = action_taken)) +
  geom_bar()+
  scale_y_continuous(labels = scales::percent)

## Training-testing Set Split

set.seed(123456)
indices.old <- sample(1:nrow(dat), nrow(dat) * 0.7)
training.old <- dat[indices.old, ]
testing.old <- dat[-indices.old, ]

## First Try: Logistic Regression

full <- glm(action_taken ~., family = binomial(link = 'logit'), data =
    ↪ training.old)
#summary(fit)
test.prob <- predict(full, testing.old, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred), reference = testing.old$
    ↪ action_taken, positive = "1")
test.roc <- roc(testing.old$action_taken ~ test.prob, plot = TRUE, print.
    ↪ auc = TRUE)

## Outliers, standardization and other adjustments

# outliers & standardization
```

```
hist(dat$applicant_income_000s)
plot(density(dat$population[dat$population < quantile(dat$population,
    ↪ 0.99)]))
lines(density(dat$applicant_income_000s[dat$applicant_income_000s <
    ↪ quantile(dat$applicant_income_000s, 0.99)]))

dat <- dat %>%
  filter(applicant_income_000s < quantile(applicant_income_000s, 0.96))
      ↪ %>%
  filter(loan_amount_000s < quantile(loan_amount_000s, 0.96)) %>%
  mutate(applicant_income = as.numeric(scale(applicant_income_000s,
      ↪ center = FALSE)),
         loan_amount = as.numeric(scale(loan_amount_000s, center = FALSE))
             ↪ ,
         population = as.numeric(scale(population, center = FALSE)),
         hud_median_family_income = as.numeric(scale(hud_median_family_
             ↪ income, center = FALSE)),
         number_of_owner_occupied_units = as.numeric(scale(number_of_owner
             ↪ _occupied_units, center = FALSE)),
         number_of_1_to_4_family_units = as.numeric(scale(number_of_1_to_4
             ↪ _family_units, center = FALSE)),
         minority_population = minority_population / 100,
         tract_to_msamd_income = tract_to_msamd_income / 100
         ) %>%
  dplyr::select(-c(applicant_income_000s, loan_amount_000s)) %>%
  dplyr::select(action_taken, everything())

hist(dat$applicant_income)
plot(density(dat$population))
lines(density(dat$applicant_income))
plot(density(dat$population[dat$population<quantile(dat$population, 0.99)
    ↪ ]))
lines(density(dat$applicant_income[dat$applicant_income<quantile(dat$
    ↪ applicant_income, 0.99)]))

## Second Try: Logistic Regression

set.seed(123456)
dat <- sample_n(dat, 1000)
indices <- sample(1:nrow(dat), nrow(dat) * 0.7)
training <- dat[indices, ]
testing <- dat[-indices, ]
full <- glm(action_taken ~., family = binomial(link = 'logit'), data =
    ↪ training)
#summary(fit)
test.prob <- predict(full, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred), reference = testing$action_
    ↪ taken, positive = "1")
```

```
test.roc <- roc(testing$action_taken ~ test.prob, plot = TRUE, print.auc
    ↪ = TRUE)

## SMOTE & L1 Regularization

# install the RSBID package
#install.packages("devtools")
#devtools::install_github("dongyuanwu/RSBID")
library(RSBID)
#training.dummy$action_taken1 <- as.factor(training.dummy$action_taken1)
#testing.dummy$action_taken1 <- as.factor(testing.dummy$action_taken1)

ptm <- proc.time()
training.bal <- SMOTE_NC(training, "action_taken")
proc.time() - ptm # running time

training.bal.dummy <- data.frame(model.matrix( ~ ., training.bal)[, -1])
testing.dummy <- data.frame(model.matrix( ~ ., testing)[, -1])
X <- as.matrix(training.bal.dummy[-1])
Y <- training.bal.dummy$action_taken1
cv <- cv.glmnet(X, Y, family = "binomial")
fit.L1 <- glmnet(X, Y, family = "binomial", alpha = 1, lambda = cv$lambda
    ↪ .min)
# confusion matrix and roc curve
test.prob <- fit.L1 %>% predict(newx = as.matrix(testing.dummy[-1]))
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
mean(test.pred == testing.dummy$action_taken1)
confusionMatrix(data = as.factor(test.pred), reference = factor(testing.
    ↪ dummy$action_taken1), positive = "1")
test.roc <- roc(testing.dummy$action_taken1 ~ as.numeric(test.prob), plot
    ↪  = TRUE, print.auc = TRUE)

## Logistic interaction

fit=glm(action_taken~.,family = "binomial",data=training.bal)

# confusion matrix and roc curve
test.prob <- predict(fit, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
mean(test.pred == testing$action_taken)
confusionMatrix(data = as.factor(test.pred), reference = factor(testing$
    ↪ action_taken), positive = "1")
test.roc <- roc(testing$action_taken ~ as.numeric(test.prob), plot = TRUE
    ↪ , print.auc = TRUE)

## interaction

fit=glm(action_taken~.+agency_code*loan_type,family = "binomial",data=
    ↪ training.bal)
```

```
# confusion matrix and roc curve
test.prob <- predict(fit, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
print("Agency_code*Loan_type")
confusionMatrix(data = as.factor(test.pred), reference = factor(testing$
    ↪ action_taken), positive = "1")
test.roc <- roc(testing$action_taken ~ as.numeric(test.prob), plot = TRUE
    ↪ , print.auc = TRUE)

fit=glm(action_taken~.+agency_code*loan_purpose,family = "binomial",data=
    ↪ training.bal)

# confusion matrix and roc curve
test.prob <- predict(fit, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
print("Agency_code*Loan_purpose")
confusionMatrix(data = as.factor(test.pred), reference = factor(testing$
    ↪ action_taken), positive = "1")
test.roc <- roc(testing$action_taken ~ as.numeric(test.prob), plot = TRUE
    ↪ , print.auc = TRUE)

fit=glm(action_taken~.+loan_type*loan_purpose,family = "binomial",data=
    ↪ training.bal)
# confusion matrix and roc curve
test.prob <- predict(fit, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
print("Loan_type*Loan_purpose")
confusionMatrix(data = as.factor(test.pred), reference = factor(testing$
    ↪ action_taken), positive = "1")
test.roc <- roc(testing$action_taken ~ as.numeric(test.prob), plot = TRUE
    ↪ , print.auc = TRUE)

#Overall

fit=glm(action_taken~.+agency_code*loan_type+agency_code*loan_purpose+
    ↪ loan_type*loan_purpose,family = "binomial",data=training.bal)

# confusion matrix and roc curve
test.prob <- predict(fit, testing, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
print("Overall")
print("Overall")
confusionMatrix(data = as.factor(test.pred), reference = factor(testing$
    ↪ action_taken), positive = "1")
test.roc <- roc(testing$action_taken ~ as.numeric(test.prob), plot = TRUE
    ↪ , print.auc = TRUE)

full <- glm(action_taken~., family = binomial(link = "logit"), data =
    ↪ training.bal.FAMD)
test.prob <- predict(full, testing.bal.FAMD, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred), reference = testing$action_
    ↪ taken, positive = "1")
test.roc <- roc(testing$action_taken ~ test.prob, plot = TRUE, print.auc
    ↪ = TRUE)

## Try: Neural Network

library(neuralnet)
ptm <- proc.time()
set.seed(123456)
NN = neuralnet(action_taken1 ~ ., training.bal.dummy, hidden = 5, linear.
    ↪ output = FALSE, err.fct = 'ce', stepmax = 1e7)
proc.time() - ptm # running time
plot(NN)
predict_NN = compute(NN, testing.dummy[-1])
test.pred <- as.numeric(ifelse(predict_NN$net.result > 0.5, 1, 0))
mean(test.pred == testing.dummy$action_taken1)
confusionMatrix(data = as.factor(test.pred), reference = factor(testing.
    ↪ dummy$action_taken1), positive = "1")
test.roc <- roc(testing.dummy$action_taken1 ~ predict_NN$net.result, plot
    ↪  = TRUE, print.auc = TRUE)

## FAMD

library(FactoMineR)
library(factoextra)
options(ggrepel.max.overlaps = Inf)

# unbalanced
FAMD1 <- FAMD(training[, -1], ncp = 8)
training.FAMD <- data.frame(FAMD1$ind$coord)
training.FAMD$action_taken <- training$action_taken
testing.FAMD <- data.frame(predict.FAMD(FAMD1, testing[, -1])$coord)
testing.FAMD$action_taken <- testing$action_taken
#balanced
FAMD2 <- FAMD(training.bal[, -1], ncp = 8)
training.bal.FAMD <- data.frame(FAMD2$ind$coord)
training.bal.FAMD$action_taken <- training.bal$action_taken
testing.bal.FAMD <- data.frame(predict.FAMD(FAMD2, testing[, -1])$coord)
testing.bal.FAMD$action_taken <- testing$action_taken

# FAMD unbalanced training set
full <- glm(action_taken ~ ., family = binomial(link = "logit"), data =
    ↪ training.FAMD)
test.prob <- predict(full, testing.FAMD, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred), reference = testing$action_
    ↪ taken, positive = "1")
```

```
test.roc <- roc(testing$action_taken ~ test.prob, plot = TRUE, print.auc
    ↪ = TRUE)

# FAMD balanced training set
full <- glm(action_taken ~ ., family = binomial(link = "logit"), data =
    ↪ training.bal.FAMD)
test.prob <- predict(full, testing.bal.FAMD, type = "response")
test.pred <- as.numeric(ifelse(test.prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred), reference = testing$action_
    ↪ taken, positive = "1")
test.roc <- roc(testing$action_taken ~ test.prob, plot = TRUE, print.auc
    ↪ = TRUE)

### Randomforest mtry1-19

library(randomForest)
test.accuracy = double(19)
test.table=array(0,c(2,2,19))
runtime=double(19)
start_time=rep(NA,19)
end_time=rep(NA,19)

for (i in 1:19) {
  start_time[i]=Sys.time()
  rf.train=randomForest(action_taken~.,data=training.bal,mtry=i)

  pred = predict(rf.train, testing)
  table=with(testing,table(pred,action_taken))
  test.table[,,i]=table
  test.accuracy[i]=(table[1,1]+table[2,2])/sum(table)
  end_time[i]=Sys.time()
  runtime[i]=end_time[i]-start_time[i]
}

test.accuracy
test.table

## Random Forest

library(randomForest)
# class of label/dependent variables should be ‘factor‘ -> classification
#training.bal.dummy$action_taken1 <- as.factor(training.bal.dummy$action_
    ↪ taken1)
rf_model <- randomForest(action_taken ~ ., data = training.bal, proximity
    ↪  = TRUE)
print(rf_model)
# type = "prob" -> gets the probability
rf_prob <- predict(rf_model, testing, type = "prob")[, 2]

#calculate RMSE
```

```
#sqrt(mean((rf_pred - testing.dummy$action_taken1)^2))

test.pred.rf <- as.numeric(ifelse(rf_prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred.rf), reference = testing$
    ↪ action_taken, positive = "1")
test.rf.roc <- roc(testing$action_taken ~ rf_prob, plot = TRUE, print.auc
    ↪  = TRUE)

# ROC for training set
#rf.roc <- roc(training.bal$action_taken, rf_model$votes[, 2], plot =
    ↪ TRUE, print.auc = TRUE)

# RandomForest based on FAMD balanced data
rf_model2 <- randomForest(action_taken ~ ., data = training.bal.FAMD,
    ↪ proximity = TRUE)
print(rf_model2)
rf_prob <- predict(rf_model2, testing.bal.FAMD, type = "prob")[, 2]

test.pred.rf <- as.numeric(ifelse(rf_prob > 0.5, 1, 0))
confusionMatrix(data = as.factor(test.pred.rf), reference = testing$
    ↪ action_taken, positive = "1")
test.rf.roc <- roc(testing$action_taken ~ rf_prob, plot = TRUE, print.auc
    ↪  = TRUE)

### XGBOOST

library(xgboost)
testing_label <- testing[, 1] #define testing label(copy from below lines
    ↪ )
training_sparse <- sparse.model.matrix(action_taken ~ . - 1, training.bal
    ↪ )
training_label <- training.bal[, 1]
train_matrix <- xgb.DMatrix(data = training_sparse, label = as.numeric(
    ↪ training_label) - 1)
testing_sparse <- sparse.model.matrix(action_taken ~ . - 1, testing)
test_matrix <- xgb.DMatrix(data = testing_sparse, label = as.numeric(
    ↪ testing_label) - 1)

params <- list(booster = "gbtree", objective = "binary:logistic", eta =
    ↪ 0.3, gamma = 0, max_depth = 6, min_child_weight = 1, subsample =
    ↪ 1, colsample_bytree = 1)
xgb.cv <- xgb.cv(params = params, data = train_matrix, nfold = 5, nrounds
    ↪  = 100)
##best iteration =

xgb1 <- xgb.train(params = params, data = train_matrix, nrounds = 100,
    ↪ watchlist = list(val = test_matrix, train=train_matrix), print_
    ↪ every_n = 10, early_stop_round = 10, maximize = F , eval_metric =
    ↪ "error")
```

```
xgb.prob <- predict(xgb1, test_matrix)
xgb.pred <- ifelse (xgb.prob > 0.5, 1, 0)
confusionMatrix(data = as.factor(xgb.pred), reference = testing_label,
    ↪ positive = "1")
test.rf.roc <- roc(testing$action_taken ~ xgb.prob, plot = TRUE, print.
    ↪ auc = TRUE)

mat1 <- xgb.importance(feature_names = colnames(training_sparse), model =
    ↪  xgb1)
xgb.plot.importance(importance_matrix = mat1)

# xgboost based on FAMD balanced training data

training_sparse <- sparse.model.matrix(action_taken ~ . - 1, training.bal
    ↪ .FAMD)
training_label <- training.bal[, 1]
train_matrix <- xgb.DMatrix(data = training_sparse, label = as.numeric(
    ↪ training_label) - 1)
testing_sparse <- sparse.model.matrix(action_taken ~ . - 1, testing.bal.
    ↪ FAMD)
testing_label <- testing[, 1]
test_matrix <- xgb.DMatrix(data = testing_sparse, label = as.numeric(
    ↪ testing_label) - 1)

params <- list(booster = "gbtree", objective = "binary:logistic", eta =
    ↪ 0.3, gamma = 0, max_depth = 6, min_child_weight = 1, subsample =
    ↪ 1, colsample_bytree = 1)
xgb.cv <- xgb.cv(params = params, data = train_matrix, nfold = 5, nrounds
    ↪  = 100)
##best iteration =

xgb2 <- xgb.train(params = params, data = train_matrix, nrounds = 100,
    ↪ watchlist = list(val = test_matrix, train=train_matrix), print_
    ↪ every_n = 10, early_stop_round = 10, maximize = F , eval_metric =
    ↪ "error")

xgb.prob <- predict(xgb2, test_matrix)
xgb.pred <- ifelse(xgb.prob > 0.5, 1, 0)
confusionMatrix(data = as.factor(xgb.pred), reference = testing_label,
    ↪ positive = "1")
test.rf.roc <- roc(testing$action_taken ~ xgb.prob, plot = TRUE, print.
    ↪ auc = TRUE)

mat2 <- xgb.importance(feature_names = colnames(training_sparse), model =
    ↪  xgb2)
xgb.plot.importance(importance_matrix = mat2)

### SVM

library(e1071)
```

```
tune.out <- e1071::tune(svm,action_taken1 ~ ., data = training.bal.dummy,
    ↪ kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5,
    ↪ 10, 100)))
# extract the best model
(bestmod <- tune.out$best.model)

svmfit = e1071::svm(action_taken1 ~ ., data = training.bal.dummy, kernel
    ↪ = "linear", cost = 0.001, scale = FALSE)

probs <- predict(svmfit, testing.dummy)
preds <- as.numeric(ifelse(probs > 0.5, 1, 0))

confusionMatrix(data = as.factor(preds), reference = as.factor(testing.
    ↪ dummy$action_taken1), positive = "1")

test.roc <- roc(testing.dummy$action_taken1 ~ probs, plot = TRUE, print.
    ↪ auc = TRUE)
```