Choose Your Own Capstone

Elizabeth

2023-09-03

Introduction

This dataset describes loan defaults, based on 36 variables. These variables include annual income, payment failure, loan amount, interest rate, amount funded, loan term, and the next payment date, among others. The data is anonymized and identifiable information is removed.

The goal of this project is to make models to predict the likelihood of someone defaulting on their loans given the loan amount and their annual income. Key steps include removing unnecessary columns and rows, removing rows containing missing values from columns used for analysis, and creating kNN and Random Forest models from the data.

Analysis

The necessary packages for the analysis include: tidyverse, caret, readr, and randomForest.

Cleaning Data

The first thing to do is load the dataset (provided in the associated GitHub repository). For the purposes of this report, the dataset was renamed to AnonLoan, instead of Anonymize_Loan_Default_data, for easier typing. To begin cleaning the data, we take a look at the first few columns of the dataset.

```
urlfile<-'Anonymize_Loan_Default_data.csv'
if(!file.exists(urlfile))
  download.file("https://github.com/eafinnigan/EdX_Capstone_CYO/raw/main/Anonymize_Loan_Default_data.csv

Anonymize_Loan_Default_data <-
  read_csv(urlfile)
```

```
## New names:
## Rows: 38480 Columns: 37
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (14): term, emp_length, home_ownership, verification_status, issue_d, lo... dbl
## (23): ...1, id, member_id, loan_amnt, funded_amnt, funded_amnt_inv, int_...
## i Use ‘spec()‘ to retrieve the full column specification for this data. i
## Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## * ‘‘ -> ‘...1‘
```

```r
rm(urlfile)

#rename data to a shorter name
AnonLoan <- Anonymize_Loan_Default_data

print(head(AnonLoan))
```

```
## # A tibble: 6 x 37
##    ...1     id member_id loan_amnt funded_amnt funded_amnt_inv term     int_rate
##   <dbl>  <dbl>     <dbl>     <dbl>       <dbl>           <dbl> <chr>       <dbl>
## 1     2      2         2         0           0               0 36 mont~     0
## 2     3 545583    703644      2500        2500            2500 36 mont~    14.0
## 3     4 532101    687836      5000        5000            5000 36 mont~    16.0
## 4     5 877788   1092507      7000        7000            7000 36 mont~     9.91
## 5     6 875406   1089981      2000        2000            2000 36 mont~     5.42
## 6     7 506439    652909      3600        3600            3600 36 mont~    10.2
## # i 29 more variables: installment <dbl>, emp_length <chr>,
## #   home_ownership <chr>, annual_inc <dbl>, verification_status <chr>,
## #   issue_d <chr>, loan_status <chr>, purpose <chr>, zip_code <chr>,
## #   addr_state <chr>, dti <dbl>, delinq_2yrs <dbl>, earliest_cr_line <chr>,
## #   inq_last_6mths <dbl>, mths_since_last_delinq <dbl>, open_acc <dbl>,
## #   pub_rec <dbl>, revol_bal <dbl>, revol_util <chr>, total_acc <dbl>,
## #   total_pymnt <dbl>, total_pymnt_inv <dbl>, total_rec_prncp <dbl>, ...
```

There is an extraneous column (originally meant as a counting column, perhaps). The first row appears to be junk data because of the December 99th value in the issue_d column. We will remove that column and row.

```r
#remove a numbering column (shown as ...1 in the data) because it isn't needed and could confuse the mo
#A similar column is already included in the dataset.
AnonLoan <- AnonLoan %>% mutate(...1 = NULL)
#Appears that the first row is junk data, (see "Dec-99" in issue_d column), so we'll remove it and look
AnonLoan <- AnonLoan[-1,]
str(AnonLoan)
```

```
## tibble [38,479 x 36] (S3: tbl_df/tbl/data.frame)
##  $ id                    : num [1:38479] 545583 532101 877788 875406 506439 ...
##  $ member_id             : num [1:38479] 703644 687836 1092507 1089981 652909 ...
##  $ loan_amnt             : num [1:38479] 2500 5000 7000 2000 3600 ...
##  $ funded_amnt           : num [1:38479] 2500 5000 7000 2000 3600 ...
##  $ funded_amnt_inv       : num [1:38479] 2500 5000 7000 2000 3600 ...
##  $ term                  : chr [1:38479] "36 months" "36 months" "36 months" "36 months" ...
##  $ int_rate              : num [1:38479] 13.98 15.95 9.91 5.42 10.25 ...
##  $ installment           : num [1:38479] 85.4 175.7 225.6 60.3 116.6 ...
##  $ emp_length            : chr [1:38479] "4 years" "4 years" "10+ years" "10+ years" ...
##  $ home_ownership        : chr [1:38479] "RENT" "RENT" "MORTGAGE" "RENT" ...
##  $ annual_inc            : num [1:38479] 20004 59000 53796 30000 675048 ...
##  $ verification_status   : chr [1:38479] "Not Verified" "Not Verified" "Not Verified" "Not Verified"
##  $ issue_d               : chr [1:38479] "Jul-10" "Jun-10" "Sep-11" "Sep-11" ...
##  $ loan_status           : chr [1:38479] "Does not meet the credit policy. Status:Fully Paid" "Charge
##  $ purpose               : chr [1:38479] "other" "debt_consolidation" "other" "debt_consolidation" .
##  $ zip_code              : chr [1:38479] "487xx" "115xx" "751xx" "112xx" ...
##  $ addr_state           : chr [1:38479] "MI" "NY" "TX" "NY" ...
##  $ dti                  : num [1:38479] 19.86 19.57 10.8 3.6 1.55 ...
##  $ delinq_2yrs          : num [1:38479] 0 0 3 0 0 0 0 0 0 0 ...
##  $ earliest_cr_line     : chr [1:38479] "Aug-05" "Apr-94" "Mar-98" "Jan-75" ...
##  $ inq_last_6mths       : num [1:38479] 5 1 3 0 4 0 0 1 0 0 ...
##  $ mths_since_last_delinq: num [1:38479] NA 59 3 72 25 NA NA NA 61 41 ...
##  $ open_acc             : num [1:38479] 7 7 7 7 8 12 5 16 15 2 ...
##  $ pub_rec              : num [1:38479] 0 0 0 0 0 0 0 0 0 1 ...
##  $ revol_bal            : num [1:38479] 981 18773 3269 0 0 ...
##  $ revol_util           : chr [1:38479] "21.30%" "99.90%" "47.20%" "0%" ...
##  $ total_acc            : num [1:38479] 10 15 20 15 25 49 9 32 44 15 ...
##  $ total_pymnt          : num [1:38479] 3075 2949 8082 2162 4206 ...
##  $ total_pymnt_inv      : num [1:38479] 3075 2949 8082 2162 4206 ...
##  $ total_rec_prncp      : num [1:38479] 2500 1909 7000 2000 3600 ...
##  $ total_rec_int        : num [1:38479] 575 874 1082 162 606 ...
##  $ last_pymnt_d         : chr [1:38479] "Jul-13" "Nov-11" "Mar-14" "Feb-14" ...
##  $ last_pymnt_amnt      : num [1:38479] 90.8 175.7 1550.3 53.1 146.8 ...
##  $ next_pymnt_d         : chr [1:38479] "Aug-13" NA NA NA ...
##  $ last_credit_pull_d   : chr [1:38479] "Jun-16" "Mar-12" "Mar-14" "Jun-16" ...
##  $ repay_fail           : num [1:38479] 0 1 0 0 0 0 0 0 0 1 ...
```

From there, duplicate entries and missing values must be found and deleted from the columns used to do analysis.

```
sum(duplicated(AnonLoan)) #no duplicate entries
```

```
## [1] 0
```

```
sum(is.na(AnonLoan)) #a lot of NAs, will check the dataset to see if they are in the columns that used
```

```
## [1] 59612
```

```
sum(is.na(AnonLoan$loan_amnt))
```

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

```
sum(is.na(AnonLoan$annual_inc))
```

```
## [1] 2
```

```
sum(is.na(AnonLoan$repay_fail))
```

```
## [1] 0
```

```
#NAs will cause problems when doing analysis. We will remove the rows that have NAs.
AnonLoan <- AnonLoan %>% drop_na(loan_amnt) %>% drop_na(annual_inc)
```

There are only 2 levels of the repay_fail column, so it will become a factor instead of a number.

```r
#there are only 2 possible outcomes with repay_fail, so we'll turn the repay_fail column from a num dat
AnonLoan <- AnonLoan %>% mutate(repay_fail = as.factor(AnonLoan$repay_fail))

#check if the mutate function worked
str(AnonLoan$repay_fail)
```

```
##  Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...
```

Training Models

First is creating the test and training sets. 20% of the data will go into the test set, because it is a large dataset and any less will make the training set harder to run. Any more and the data may become overtrained.

```r
set.seed(29)
test_index <- createDataPartition(y = AnonLoan$repay_fail, times = 1, p = 0.2, list = FALSE) # create a
test_set <- AnonLoan[test_index,]
train_set <- AnonLoan[-test_index,]
```

The models trained will be k-Nearest Neighbors and Random Forest. k-Nearest Neighbors can handle multiple dimensions in conditional probabilities, and Random Forest can model decision making processes while still being accurate and flexible. Cross-validation is needed for both to become most accurate.

K-Nearest Neighbor First is the k-Nearest Neighbor.

```r
#train the model
train_knn <- train(repay_fail~annual_inc + loan_amnt, method = "knn", data = train_set)
#make a prediction
y_hat_knn <- predict(train_knn, test_set, type = "raw")
#see accuracy of model
confusionMatrix(y_hat_knn, test_set$repay_fail)$overall[["Accuracy"]]
```

```
## [1] 0.8460239
```

The accuracy is 0.8460239 without cross-validation. To make sure that we are not overtraining and the accuracy is accurate, we will use cross-validation.

```r
#train model with cross-validation - see if we can get the accuracy up
set.seed(29)
train_knn_test <- train(repay_fail~annual_inc + loan_amnt, method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(9, 71, 2)))

#confusion matrix with best accuracy
confusionMatrix(predict(train_knn_test, test_set, type = "raw"), test_set$repay_fail)$overall["Accuracy
```

```
##  Accuracy
## 0.8484927
```

With cross-validation, the accuracy is now 0.8484927, which is more accurate.

Random Forest Next model is the Random Forest model.

```
set.seed(29)
train_rf <- randomForest(repay_fail~annual_inc + loan_amnt, data=train_set)
confusionMatrix(predict(train_rf, test_set), test_set$repay_fail)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8380977
```

The accuracy is 0.8380977 without cross-validation. Will use cross-validation to improve the accuracy and prevent overtraining.

```
set.seed(29)
train_rf_test <- train(repay_fail~annual_inc + loan_amnt,
                      method = "Rborist",
                      tuneGrid = data.frame(predFixed = 2, minNode = c(3, 50)),
                      data = train_set)
#confusion matrix with best accuracy
confusionMatrix(predict(train_rf_test, test_set), test_set$repay_fail)$overall["Accuracy"]
```

```
## Accuracy
## 0.847843
```

The accuracy is now 0.847843 with cross-validation, which is slightly worse than the kNN model.

Results

The results of the kNN model are:

```
train_knn_test$finalModel
```

```
## 71-nearest neighbor model
## Training set outcome distribution:
##
##     0     1
## 26120  4661
```

The results of the Random Trees model are:

```
train_rf_test[["finalModel"]][["validation"]][["confusion"]]
```

```
##       0  1
## 0 26105 15
## 1  4654  7
```

Accuracy of kNN model: 0.8382277 without cross validation, and 0.8484927 with.

Accuracy of Random Trees model: 0.8382277 without cross-validation, and 0.847843 with.

Overall, the k-nearest neighbor model performed better with accuracy, although Random Trees was very close. Both models required cross-validation to increase their accuracy. If cross-validation cannot be done, then Random Trees gives a higher accuracy than kNN.

Conclusion

There is a high correlation between annual income, loan amount, and whether someone managed to pay off their loans.

The potential impact of this is the ability to tell whether or not someone will default on their loans, which could give debitors the ability to make good investments with their money.

The limitations of this model include the fact that not paying loans is very rare, so most predictions of loan default will be false positives, as shown by the results of the Random Trees model.

References

The dataset is from https://www.kaggle.com/datasets/joebeachcapital/loan-default, uploaded by Joakim Arvidsson (username joebeachcapital).

Help with loading the CSV file from GitHub was done with answers from here: https://stackoverflow.com/questions/14441729/read-a-csv-from-github-into-r.