Bank Client Defaults

Sudhakar Naidoo

2022-08-11

Overview

The following report is based on data from a Kaggle competition (American Express - Default Prediction) that was ongoing when this report was completed. The competition relates to identifying customers who will default on their debt repayments.

The data also includes a file called train_labels.csv which identifies clients who have defaulted and those who haven't.

I used the train set and extracted just 10 000 rows from the more than 500 000 rows available. Due to the time required to train the data and hardware requirements, I chose a 50/50 split. I also excluded 180 columns which were predictors. I would prefer an 80/20 split if I could use all the data available. This is to help ensure that the algorithm learns as much as possible before testing and deploying a machine learning model. Predictions didn't take long and so I felt that a 5000 row test set would better evaluate the accuracy of my model.

The variables fall into the following categories:

D_* = Delinquency variables S_* = Spend variables P_* = Payment variables B_* = Balance variables R_* = Risk variables

Method

I extracted the first 5000 rows using the code below.

```
# reading 5 000 lines from train_data to use as train data
train <- read.csv("train_data.csv", nrows = 5000)
```

I added column names to the train set. The columns include the customer identification number, the date and eight predictors relating to client bank transactions.

```
# Add column names
colnames(train) <- c("customer_ID","S_2", "P_2", "D_39", "B_1", "B_2", "R_1", "S_3", "D_41",
"B_3")
```

I deleted the other 180 columns which included many more transaction related predictors.

```
#delete unused columns to predict on the first 10 predictors
train <- subset(train, select = c("customer_ID","S_2", "P_2", "D_39", "B_1","B_2", "R_1", "S_
3", "D_41", "B_3"))
```

"train1" below represents the train set file joined by "customer_id" to the train_label.csv file so that it includes the outcome ie. the "target" which equals either 1 for a default or 0 otherwise.

I transformed the date from a character type to a date type.

```
#transform to date
train1[,2] <-as.Date(train1[,2])
```

I also transformed all other columns used. The factor relates to the “target” column ie. 1 or 0.

```
#transform to factor and numeric
train1[,3] <- as.numeric(train1[,3])
train1[,4] <- as.numeric(train1[,4])
train1[,5] <- as.numeric(train1[,5])
train1[,6] <- as.numeric(train1[,6])
train1[,7] <- as.numeric(train1[,7])
train1[,8] <- as.numeric(train1[,8])
train1[,9] <- as.numeric(train1[,9])
train1[,10] <- as.numeric(train1[,10])
train1[,11] <- as.factor(train1[,11])
```

I confirmed the class of the target.

```
#Check that "target" is a factor
class(train1$target)
```

```
## [1] "factor"
```

I identified 871 NAs in the train set.

```
#Check for NAs
no_nas <- ifelse(is.na(train1), 0, train1)
sum(is.na(train1))
```

```
## [1] 871
```

I removed the rows with the NAs to ensure the model only predicts on a client when all predictors are available.

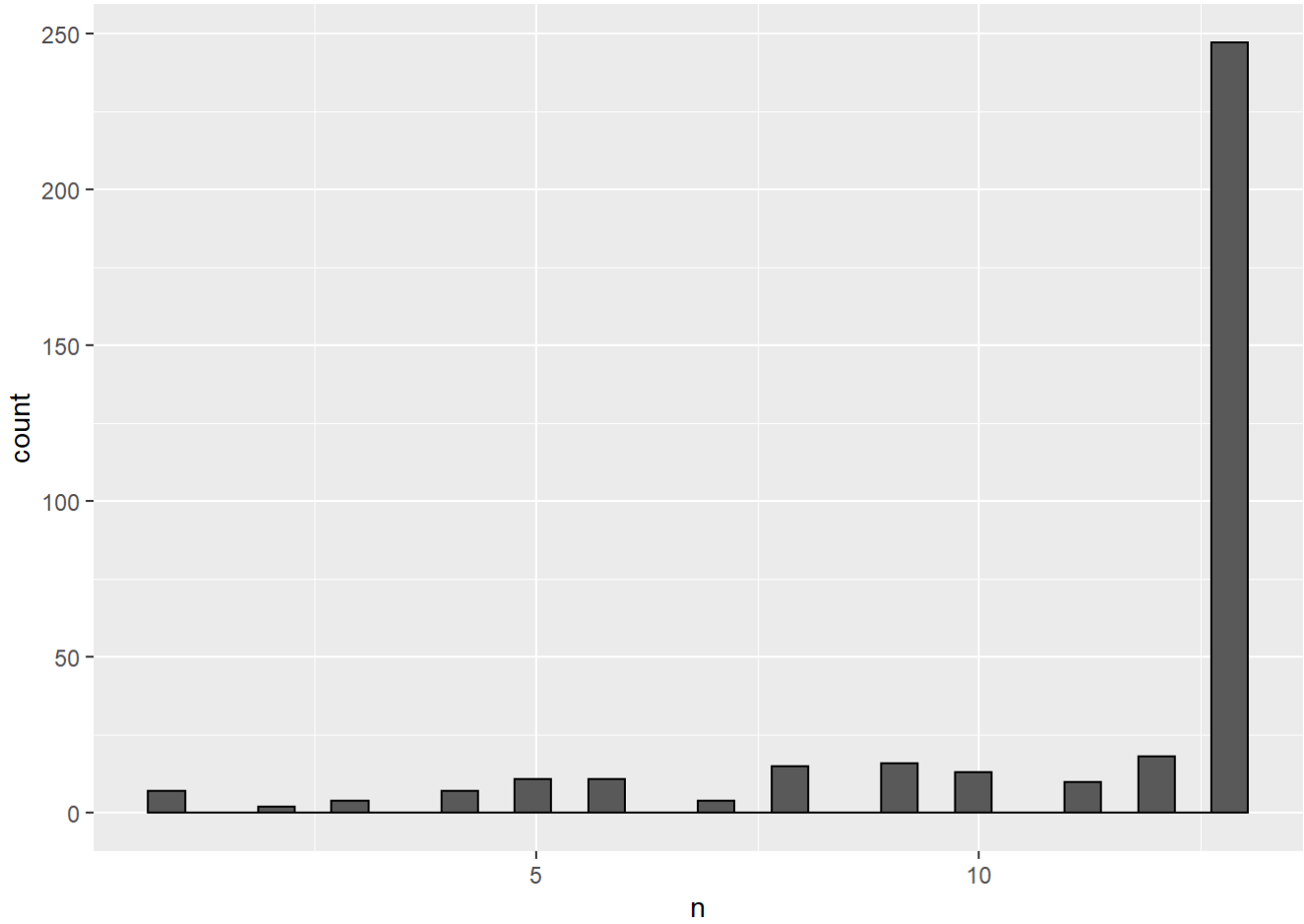
```
# remove rows with NA's to only predict on clients when there all predictors are available
train1 <- train1[complete.cases(train1),]
```

I computed the number of unique customers in the dataset and found 365 unique customers.

```
# number of unique customers
unique_customers <- train1%>%
  group_by(customer_ID)%>%
  summarize(n=n())%>%
  arrange(desc(n))%>%
  count()
unique_customers
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   365
```

The histogram below plots the number of statement dates per client. It shows that most clients have about 13 statement dates.



I read 5000 more lines from the available data to use as the test set. I skipped the first 20 000 rows.

```
# reading 5000 lines from train_data to use as test data
test <- read.csv("train_data.csv", nrows = 5000, skip = 20000)
```

I added column names for the test set as done for the train set.

```
# Add column names
colnames(test) <- c("customer_ID","S_2", "P_2", "D_39", "B_1", "B_2", "R_1", "S_3", "D_41",
"B_3")
```

I deleted columns not used for test set as done for the train set.

```
#delete unused columns
test <- subset(test, select = c("customer_ID","S_2", "P_2", "D_39", "B_1","B_2", "R_1", "S_3"
, "D_41", "B_3"))
```

“test1” represents the test file joined with the train_label.csv file the same way it was done for the train set.

```
#transform to date
test1[,2] <-as.Date(test1[,2])
```

Again, I transformed the data the same way I did for the train set.

```
#transform to factor and numeric
test1[,3] <- as.numeric(test1[,3])
test1[,4] <- as.numeric(test1[,4])
test1[,5] <- as.numeric(test1[,5])
test1[,6] <- as.numeric(test1[,6])
test1[,7] <- as.numeric(test1[,7])
test1[,8] <- as.numeric(test1[,8])
test1[,9] <- as.numeric(test1[,9])
test1[,10] <- as.numeric(test1[,10])
test1[,11] <- as.factor(test1[,11])
```

I checked that the target is a factor type in the test set which was required to train the model.

```
#Check that "target" is a factor
class(test1$target)
```

```
## [1] "factor"
```

I checked for NAs and found 1070 of them.

```
#Check for NAs
no_nas <- ifelse(is.na(test1), 0, test1)
sum(is.na(test1))
```

```
## [1] 1070
```

Again, I removed the rows with NAs to ensure the model only predicts on clients with the chosen predictors available.

```
# remove rows with NA's to only predict on clients when there all predictors are available
test1 <- test1[complete.cases(test1),]
```

I checked the final dimensions of the train and test sets and found they were still approximately a 50/50 split after removing rows with NAs.

```
# the dimensions of the train and test sets
dim(train1)
```

```
## [1] 4131   11
```

```
dim(test1)
```

```
## [1] 3980   11
```

I decided to use multiple models in an ensemble to get predictions that are more accurate than most models used in the ensemble. This is due to the nature of how an ensemble works by calculating row means across a number of models for each statement date. These row means represent the mean of the output for the different models, which is either a zero or one for each statement date. If more models output one than zero, then greater than 50% of models believe the client will default and the ensemble will predict a default.

Results

The accuracy of the ensemble was impressive at close to 82%. As their was so much data available, I tested the model two more times by randomly choosing 5000 rows at a time from the available data. I have included these datasets as test_data1.csv and test_data2.csv.

The model maintained an accuracy of between 80% and 82% when tested with all three test sets.

The accuracy for each model and the mean accuracy is computed below:

```
accuracy <- colMeans(predictions == test1$target)
accuracy
```

```
##       glm       lda        nb svmLinear       knn  gamLoess  multinom       qda
## 0.8218593 0.8201005 0.8170854 0.8218593 0.7952261 0.8336683 0.8218593 0.8027638
##        rf
## 0.8193467
```

```
mean(accuracy)
```

```
## [1] 0.8170854
```

The ensemble accuracy is computed below:

```
Model_Answers <- rowMeans(predictions == "1")
y_hat <- ifelse(Model_Answers > 0.5, "1", "0")
mean(y_hat == test1$target)
```

```
## [1] 0.8241206
```

Conclusion

The model has done well to predict customers who default. However, it would be nice to use the other 180 predictors and the full dataset and compare to the results above.