

Deduping Time-Varying Redundant Transactions in Commercial Lending

Patralekha Bhattacharya, Jigyasa Analytics

ABSTRACT

Deduping duplicate transactions is an important step when building a predictive model, and models that are built to create credit risk scorecards in the commercial lending industry are no exception. What is specific to this industry, however, is that many duplicate transactions occur because the borrower splits the loan amount into multiple parts and asks for the loan on a piecemeal basis in order to increase the chances of loan approval. These individual loans appear in the dataset as separate records—even though they are actually part of the same transaction and only one of them should be kept. Since borrowers care more about paying off their larger loans on time than the smaller ones, we prefer to keep the transaction with the largest transaction amount when removing duplicates.

A commonly used method is to only remove the duplicates that appear within the same calendar quarter. This method is very easy, and one can use a quick sort to only keep the transactions with the highest transaction amount. However, the drawback of this method is that transactions that may have taken place just one day apart (e.g., March 31st and April 1st) will not be candidates for deduping as they do not belong to the same calendar quarter.

In this paper, we present a different method of deleting duplicate observations when they are within 90 days apart. Since the time-range we consider depends on the date of each transaction, this method is more complex and may need multiple iterations. However, if done correctly, this method will be more accurate and will remove more duplicate transactions correctly than the above calendar quarter deduping method.

INTRODUCTION

Practitioners in the predictive modeling industry often have to handle duplicate records in their datasets.

Sometimes, these records may be exact duplicates, which means that all the variables take the same value in two or more rows. At other times, they may be partial duplicates, where most variables are exactly the same, but some variables differ between rows.

There are many reasons why duplicate transactions may be observed in the data. These include (but are not limited to):

- (i) Data entry errors where the same data is inaccurately entered multiple times. In this case, the duplicated observations may be exact duplicates of each other.
- (ii) Storing the same data in different tables, which are then appended or merged. For instance, if data on purchases and returns are kept in two separate tables, there may be two rows for the same transaction when the two tables are merged: one row for when the purchase occurred and another row for when the purchase was canceled and the product returned. In this case, the data in the two rows are not exact duplicates (as the status of the transaction is different), but all other variables would still be exactly the same.
- (iii) Sometimes, duplicate data may result from multiple transactions by the same individual or business. Once again, these will be partial duplicates. While the buyer and seller information are exactly the same, the product purchased, the purchase amount, and the transaction status may be different between rows.

Using duplicate observations during the modeling process will incorrectly add extra weight to those observations, which can bias the results of the models. Sarracino and Mikucka (2017) performed several Monte Carlo simulations to understand the bias that arises when there are duplicates in the data. They found that an increase in the number of duplicates in the dataset increases the bias in the regression coefficients. Moreover, the risk of obtaining biased estimates increases with the share of duplicates in the data. Also, when the duplicates constitute the same share of the data, the risk of obtaining biased

estimates is higher for quintuplets than for doublets. Both researchers and practitioners in the industry agree that, in most situations, it is preferable to remove duplicate observations before the modeling process.

However, which data to delete and the preferred removal method for duplicate transactions vary depending on the situation at hand.

In case of data entry errors, it makes sense to keep the more accurate or complete entry and remove the others.

In the case of a subscription-based service, for instance, duplicate data may arise if the customer subscribes and then cancels their subscription a few months later. In this scenario, which is similar to our earlier example where a product is first purchased and later returned, we may only be interested in the customer's final status (i.e., whether they canceled, returned, or kept the product). If so, it makes sense to use the most recent information for this customer (i.e., to delete the earlier transaction).

There are several other reasons why duplicate transactions may arise in the commercial lending industry. Sometimes, the same borrower may borrow money on different dates, for different pieces of equipment, or for pieces of equipment that are being delivered to different locations. While this happens for small borrowers, too, it is even more prevalent for large borrowers, who may purchase several different types of equipment and, therefore, may make several different transactions requesting small amounts of money.

In fact, sometimes, even if the borrower is borrowing money for the same equipment, it can be easier to get the loan approved if the amount of credit requested is lower. Therefore, they may decide to divide up the amount requested into two or more parts and ask for them on a piecemeal basis. When they do split up the requested loan amounts in this manner, the multiple requests usually occur over a short period of time (such as within a three-month window).

In these situations, duplicate transactions should be removed for the following reasons.

- For small borrowers: if one transaction is being split up into multiple transactions, keeping all of them in the data will lead to double counting.
 - In this case, it makes sense to sum up the smaller transaction amounts to get the correct amount requested for the borrower.
- For large borrowers: If a large borrower has many transactions, each transaction will be a separate record in the data. Much of the data used for modeling will be exactly the same for all of these records, and therefore, keeping all of the records in the dataset will significantly bias the model towards the data of that one large borrower.
 - Once again, removing duplicate transactions by the same borrower makes sense.
 - Another point to consider about the larger borrowers is that, since these borrowers have an existing long-term relationship with the lender (and may be a valuable customer for the lender), they may be lax about repaying smaller loans back on time but be much more careful about timely repayment of larger loans. Therefore, with multiple transactions, if we keep the transaction with the smaller loan amount, and the borrower does not pay that loan off but instead pays off the larger loan amount, we may inaccurately classify the borrower as a defaulter when he is not. Therefore, in this situation, it makes sense to keep the transaction with the highest amount, which the borrower has likely been much more careful about paying off.

For the above reasons, the usual practice is to remove duplicate transactions within a specified window. The most commonly used method is to separate the data into calendar quarters and then remove duplicates within the same calendar quarter, leaving duplicate transactions in different calendar quarters.

This method of deduping is simple. We need only look at duplicate transactions by the same borrower within a specified calendar quarter and keep the transaction with the largest transaction amount, a task that can be easily accomplished by a quick sort. However, the drawback of separating the data into calendar quarters is that transactions that have taken place just one day apart (e.g., March 31st and April 1st) will not be candidates for deduping. We cannot, in the vast majority of cases, expect the ease of

removing duplicates during the modeling process to be at the forefront of a borrower's mind when they are scheduling their transactions. Therefore, in most situations, using the same calendar quarter deduping method will fail to remove all of the duplicate transactions that should be removed.

This problem is visualized in the following figure (Figure 1):

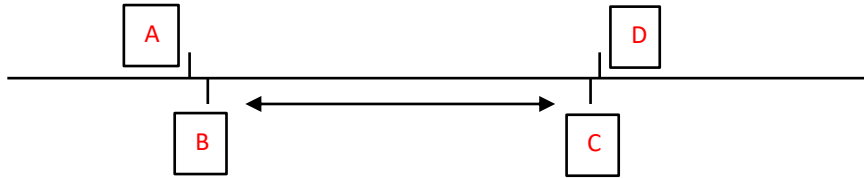


Figure 1: Calendar Quarter Deduping

Consider a borrower with four transactions: A, B, C, and D. Let us assume that the period between B and C is one calendar quarter. With the calendar quarter approach, B and C are considered to be part of the same transaction. However, A and B are considered to be different transactions even though they are much closer together in time than B and C are from each other. Similarly, even though transactions C and D occur very close together in time, they are also considered different transactions with the calendar quarter approach.

In this paper we present a method of deleting duplicate observations when they are within 90 days of each other. This method is more complicated than the calendar quarter method. However, it removes duplicate observations more accurately.

In this article, we present this methodology in the context of the commercial lending industry. However, the same method can be extended to any other situation where the removal of time-varying duplicates is required.

DATASET

A typical commercial lending application dataset consists of borrowers who have applied for one or more loans from a commercial lender. The dataset contains the following variables:

- **Borrower_id**: a unique ID identifying the borrower applying for the loan.
- **Transaction_id**: This variable uniquely identifies each loan application. A borrower may have multiple loan applications, each with a different transaction_id. The dataset is unique by transaction_id.
- **Transaction_date**: The date the borrower applies for the loan.
- **Requested_Amount**: This is the amount of loan requested by the borrower during a particular transaction.

A SIMPLE METHOD OF DEDUPING BY CALENDAR QUARTER

A simple SAS code for deduping the duplicate transactions (within each calendar quarter) in this dataset could be written as follows:

```
*** Assume the dataset called "base" contains the above variables;
data base1;
set base;
month_trans_date = month(Transaction_date); ** calculate month of
transaction_date;
year_trans_date = year(Transaction_date); *** calculate year of
transaction date;
```

```

*** create calendar quarter from transaction date;
If month_Trans_date >= 1 and month_trans_date <= 3 then qtr_trans_date = 1;
If month_Trans_date >= 4 and month_trans_date <= 6 then qtr_trans_date = 2;
If month_Trans_date >= 7 and month_trans_date <= 9 then qtr_trans_date = 3;
If month_Trans_date >= 10 and month_trans_date <= 12 then qtr_trans_date =
4;
yrqtr = cats(year_trans_date, "0", qtr_trans_date);
run;

*** sort by borrower_id calendar_quarter and descending Requested_Amount;
proc sort data = base1; by borrower_id yrqtr descending RequestedAmount;
run;

*** For each borrower_id and calendar_quarter, we remove duplicates to keep
the row with the largest Requested_Amount;
proc sort data = base1 out = base2 dupout = dups nodupkey equals; by
borrower_id yrqtr; run;

```

In our simulated dataset, the above method removed many duplicate observations.

However, there are several observations in our dataset where the same borrower requests loans a few days or weeks later, but the duplicates are not removed because the transactions do not appear in the same calendar quarter.

A MORE GENERAL AND TIME-VARYING METHOD OF DEDUPING

The above method almost inevitably leaves many observations in the dataset that are part of the same transaction. In fact, in the above dataset, if $x\%$ of the applications were deduped using the calendar quarter method, approximately $(x + 30)\%$ should have been deduped (as they were made less than 90 days apart by the same borrower_id) but were not due to the limitations of the calendar quarter method. We have seen datasets where this figure is as high as $(x + 60)\%$. It is imperative that we remove these duplicate observations if we want to prevent biases during the modeling stage.

As stated above, the improved deduping method is more complicated, as we have to compare every transaction by a borrower with every other transaction by the same borrower before we can decide which ones to remove. When removing duplicates in this manner, we must be mindful of a few things:

First, with both methods, there are multiple ways of deduping the data when we do not have any additional constraints.

Let's consider a simple case where the same borrower sends in three application requests on March 15th, 2022, April 1st, 2022, and June 25th, 2022. With the calendar quarter method, either the April 2022 or the June 2022 transactions would be removed, as they belong to the same calendar quarter. The March 2022 transaction, however, would never be removed, as it is the only one in the first calendar quarter of 2022. Therefore, there are two ways of deduping the data, using the calendar quarter approach: (i) one in which the March and April transactions would be kept, and the June transaction would be dropped, and (ii) the second in which the March and June transactions would be kept and the April transaction dropped. In both cases, two out of these three transactions would remain in the dataset.

With the 90-day method, since March and April are less than 90 days apart, and April and June are less than 90 days apart, we again have two deduping options. The first is to remove both the March and the June transactions and keep only the April 2022 transaction. The second is to remove only the April 2022 transaction (as, in that case, the March and June transactions are more than 90 days apart). Without any other considerations, both these methods would be logically correct, and once again, we have two ways of deduping the data using the 90-day approach.

Secondly, using an additional constraint (such as preferring the observation with the higher requested amount) will reduce the number of deduping options we have. Let us continue with the above example to help us understand this.

Let's now suppose that the requested amounts for March, April, and June of 2022 are \$6000, \$5000, and \$4000, respectively, and we always want to keep the transaction with the highest requested amount. (As explained earlier, this is often good practice in commercial lending, as the borrower has a higher interest in paying off loans where the requested amount is high and may, in fact, have an unofficial understanding with the lender where they pay more than the required amount on the high loans while letting the lower value loans lapse for some time). In the above scenario, this means that, while deciding which duplicates to keep, we will prefer the records with higher requested amounts. With the calendar quarter method, we would remove the June 2022 transaction from this dataset as the requested amount in June 2022 is lower than that in April 2022, and they both belong to the same calendar quarter. Therefore, we will be left with only the March and April transactions, and there is only one way of removing duplicates with the calendar quarter approach now as opposed to the case in the previous example, where we could do this in two ways (with either pulling the April transaction or the June transaction).

With the 90-day approach, since the April transaction has a lower requested amount than the March transaction, we would remove the April transaction from our dataset, and the 90-day approach would result in us keeping the March and June transactions. Once again, in this case, we are left with only one way of removing duplicates because of the additional constraint.

Lastly, some transactions that are removed using the calendar quarter approach will not be removed when using the 90-day approach, and vice versa.

Figure 1 shows an example of a situation where transactions deduped using the 90-day approach may not be deduped using the calendar quarter approach. This happens when the transactions are only a few days apart but do not belong to the same calendar quarter.

However, the opposite situation may also arise, as we saw in the example above, with March, April, and June transactions with transaction amounts of \$6000, \$5000, and \$4000, respectively. With the calendar quarter approach, we would keep the March 2022 and April 2022 transactions and remove the June 2022 transaction (since it belongs to the same calendar quarter as the April 2022 transaction). With the 90-day approach, however, we would remove the April 2022 transaction (since it is within 90 days of the March 2022 transaction) and keep the March 2022 and June 2022 transactions.

In this case, the two methods will remove different transactions. However, since the March 2022 and April 2022 transactions are closer in time, they are more likely to be part of the same transaction. Therefore, the 90-day approach is more likely to remove the correct transactions than the calendar quarter approach.

METHODOLOGY TO REMOVE TIME-VARYING DUPLICATES

The methodology used to remove time-varying duplicates compares each borrower's transaction with every other transaction by the same borrower within 90 days. We do this by joining the dataset with itself and comparing each pair of transactions by the same borrower. Of these compared transactions, we filter out the ones that are only 90 days apart, and then we remove the one with the lower requested amount.

However, too many transactions may be removed if we stop at the above step. In our earlier example of transactions in March, April, and June with requested amounts of \$6000, \$5000, and \$4000, respectively, the April transaction will be removed as it has a lower transaction amount than the March transaction. The June transaction will also be removed as it has a lower amount than the April and March transactions. Since we want to keep the March transaction (as it has the highest transaction amount), we must remove the April transaction. However, once the latter has been removed, there is no need to drop the June transaction. Therefore, that should be returned to our dataset in the following steps.

To do this programmatically, we have to devise a method that accounts for this scenario and allows us to keep transactions that do not need to be dropped. We can do this with the help of a SAS macro that drops transactions iteratively.

The logic of the SAS macro is as follows: For each borrower_id, we first find the transaction with the largest transaction amount. This transaction will never be deduped, as we always want to give preference to keeping the larger transactions. After this transaction is kept, all transactions with the same borrower_id within 90 days of it are immediately deduped. After dropping these other transactions within 90 days of this largest transaction, we once again try to identify the largest transaction among those remaining for the borrower_id and repeat the process, keeping that transaction and again deduping the ones within 90 days of it. We continue in this manner, keeping and dropping transactions iteratively until no duplicate transactions are left to remove.

PROS AND CONS OF EACH METHOD

Removing duplicates by calendar quarter is relatively simple to implement, whereas the 90-day method is much more complex. Our macro makes it simple for the case when we wish to keep the transactions with the higher requested amount; however, when working with real-world datasets, situations with a different constraint (where the transactions that need to be kept have a characteristic other than the highest requested amount) are likely to arise. In these situations, the macro must be modified, and the logic must be thought through and altered accordingly.

Removing duplicates using the calendar quarter method will sometimes remove duplicates that are further apart in time while not removing transactions that are closer together just because they belong to different calendar quarters.

Removing duplicates using the 90-day approach will always remove duplicates within the same calendar quarter; if two transactions by the same borrower_id belong to the same calendar quarter, both the calendar quarter approach and the 90-day approach will always remove one of those transactions. However, as shown above, the specific transaction that is removed may differ between the two methods.

Proponents of the calendar quarter approach argue that since the bureau data is archived and available quarterly, using the calendar quarter deduping method is enough, and there is no need for the 90-day deduping method. However, while it is true that the bureau data is usually available quarterly, there are often gaps in the data. In our experience, we have found that borrower information frequently has to be obtained from other adjacent quarters (different from the quarter in which the transaction took place) to fill those gaps. Therefore, the 90-day approach is preferable as it can be used in all these situations.

CONCLUSION

In this paper, we present a time-varying method of deleting duplicates. While we have focused on the commercial lending industry, this method is helpful in any situation where we want to remove duplicates that occur close to each other in time. We used 90 days as the period to look for duplicates. However, this methodology can be extended to any other period, such as 30, 60, 120, or 180 days, etc., as the user desires.

In our datasets, we found that the 90-day method of dropping duplicates could find and remove many more duplicates than the calendar quarter method. The calendar quarter method often misses duplicates that are close to each other in time but belong to different calendar quarters. The 90-day method, more accurately removes duplicate transactions that are close to each other in time.

REFERENCES

Sarracino, F., & Mikucka, M. (2017). Bias and efficiency loss in regression estimates due to duplicated observations: a Monte Carlo simulation. *Survey Research Methods*, 11(1), 17–44.
<https://doi.org/10.18148/srm/2017.v11i1.7149>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Patralekha Bhattacharya
Jigyasa Analytics, LLC

E-mail: pbhattacharya@jigyasaanalytics.com , patrab@gmail.com
Web: www.jigyasaanalytics.com