Identifying Recurring and Missed Payments in Bank Statements: A Graph-Based Approach

Jason Young

Practicum Thesis for Master of Science in Computational Analytics

Sponsored by:
Georgia Institute of Technology
and
Nova Credit Inc.

July, 2024

Overview

This paper explores the question of how to identify recurring payments, missed payments, and late payments from an individual's bank statement. This problem is highly relevant for a variety of commercial applications, such as credit application decisioning or determining whether to grant rental offers. The importance of accurately classifying such transactions has grown considerably alongside the growth of APIs that facilitate the sharing of consumer-permissioned financial data.

The paper is structured as follows:

1. **Dataset Description**
   A brief overview of the data is provided and how it was filtered and sampled for this analysis.
2. **Feature Exploration and Transformation**
   This section discusses the motivation for mapping the transaction date onto a circular space. It then explores a 3-dimensional visualization of the transaction space, which will build our intuition for the payment-grouping algorithm.
3. **Identifying Recurring Payment Groups Using a Graph-Based Approach**
   3.1 describes the mechanics of the author's unique graph-based algorithm to identify recurring payment streams.
   3.2 compares a set of selected payment grouping methods using Normalized Mutual Information (NMI).
   3.3 explores the tradeoff between classifying a higher number of recurring payments and the false positive rate

4. **Identifying Late and Missing Payments**
   4.1 describes the mechanics of a rules-based algorithm to late and missing payments.

4.2 looks at the distribution of late payments and how the late-payment cutoff threshold impacts the percentage classified as late.
4.3 offers a qualitative discussion of the results of the missed payment algorithim.

Throughout this paper, the author will focus on monthly payments for simplicity. However, all approaches covered could easily be extended to bi-weekly or quarterly recurring payments.

1. Dataset Description

This analysis uses a confidential dataset provided by the Nova Credit Inc, the project sponsor. The original dataset consists of 247,269 anonymized transactions from the bank statements of 150 individuals, occurring over a 3-year period. Individuals are referred to here as “applicants”, since the project sponsor commonly uses this data to assist in making credit application decisions.

The raw dataset is low-dimensional, with only 3 relevant features;

- Date: The day, month, and year of a transaction, not including time-stamps.
- Amount: The dollar value of a transaction, where negative values represent expenses and positive values income.
- Description: A brief description of the transaction, scrubbed to remove Personally Identifiable Information (PII).

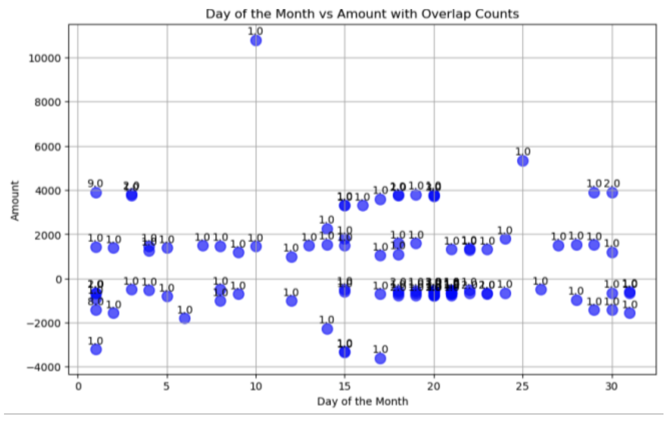
Each transaction also contains an account field to identify the applicant. However, since all algorithms in this paper are applied to only one applicant at a time, this field is only used to subset individuals as unique inputs for the algorithms.

Although the full dataset was used to explore and create the algorithms, all analysis presented in this paper was performed on a random sample of 5 applicants. The smaller sample size was necessary to perform the manual evaluations in sections 3 and 4. The data was further filtered to exclude transaction amounts less than $200 or greater than $15,000, since these small and extra-large transactions are unlikely to be relevant to the problem of credit application scoring.

With 866 transactions remaining, the final sample is nonetheless sufficient to demonstrate the efficacy of the algorithms and to identify recurring and missed payments.

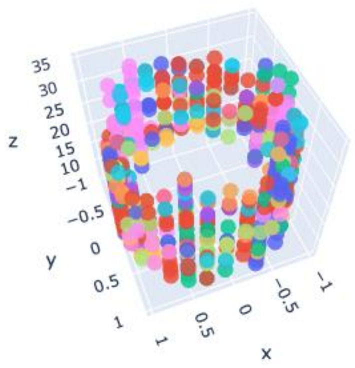
2. Feature Exploration and Transformation

An initial exploration of the data investigated grouping together payments with similar amounts and nearby days of the month, as seen in the scatterplot below.



Here, we can see clusters of overlapping payments that are likely to belong to the same recurring group. However, one key limitation emerges. In this space, a payment made on the 31st is far from a payment made on the 1st, even though they may only be 1 day apart.

To solve this issue, a function was created to map each day of the month onto a circle. Building on this concept, we can visualize the cyclicality of the transaction dates. The figure below shows one way to visualize the transaction space as a 3-dimensional cylinder.



Each point represents a single transaction, plotted in time. For a fixed z-value, each day of the month (the 1st through he 31st) is mapped to a circle on the x and y axis. One full circle represents the passage 31 days within a month. Each month (circle) is stacked vertically on the z-axis, so the bottom circle is the beginning month of transactions and the top circle is the last.

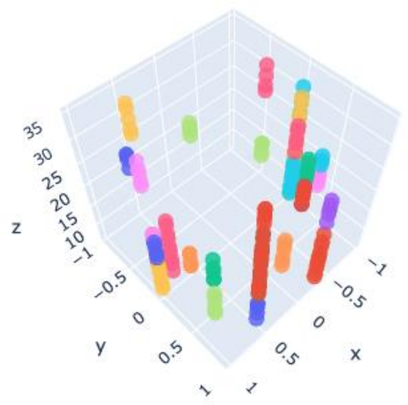
Put more concisely, time is shown as a spiral or coil from the bottom of the cylinder to the top.

Representing transactions in this space allows us to see that recurring monthly transactions will tend to form a vertical line, but with some variation on the exact day when they occur. In 3D plot above, the transaction's color represents its description (which is often a good proxy for its payment grouping). Looking closely, we can see that some transactions that share the same description/color do follow this monthly pattern and form a vertical line.

With this 3D observation of the transaction space, two additional issues arise. First, it is clear that description alone is not a strong identifier of recurring transactions. A given description (such as "check" for cashing a check) may apply to a variety of different payments. Additionally, the description for a given payment stream might change over time.

Second, we will need a method that can identify recurring transactions when the monthly cadence is not exact and where the transaction day can be slightly earlier or later than their usual pattern.

These issues will motivate the recurring payment identification algorithm described in the next section. The new 3D plot below shows the same transaction space using categories produced by the algorithm and filtering out one-off transactions.



3. Identifying Recurring Payment Groups Using a Graph-Based Approach

3.1 Description of the Graph-Based Algorithm

My approach to identify recurring payments is to produce a graph of transactions where each connected component of the graph represents a grouping of recurring payments.

To do this, I define a binary adjacency Matrix G, where $G_{i,j} = 1$ if transaction i should belong to the same payment stream as transaction j. In other words, I evaluate each distinct pair of transactions to determine if they should be connected.

Each entry $G_{i,j}$ of the adjacency matrix is conditional on the transaction pair having a similar dollar amount, and a time-pattern consistent with a recurring monthly cadence.

To test the dollar amount similarity, I will define another square binary matrix A, where $A_{i,j} = 1$ if the dollar amount of transaction i, $amount_i$ is similar to the dollar amount transaction j, $amount_j$. For simplicity, a \$50 threshold was used, meaning that:

$A_{i,j} = 1$ when $(amount_j - 50) < amount_i < (amount_j + 50)$

Next, to test the time-pattern consistency between transactions, I define another square binary matrix T, where $T_{i,j} = 1$ if transaction i's date falls within the expected monthly cadence before or after transaction j's date. More specifically, $date_i$ must occur within 3 months before or after $date_j$ (3 months was chosen instead of 1 month to allow the algorithm to capture skipped payments discussed in section 4). Additionally, $date_i$ must be near the same day of the month of $date_j$, for example if the day threshold is 6, and $date_i =$ the 2nd of the month, then $date_j$ must be between the 27th and the 8th.

Combining the information from matrix A and matrix T, we can define each element of the adjacency matrix G as follows:

$G_{i,j} = 1$ when $(A_{i,j} = 1 \textit{ and } T_{i,j} = 1)$
$G_{i,j} = 0$ otherwise

With adjacency matrix G populated, we can build a graph and extract the connected components with Python's NetworkX. Finally, each transaction is labeled as a unique grouping of recurring payment streams. All non-recurring payments (non-connected components) are put into their own group.

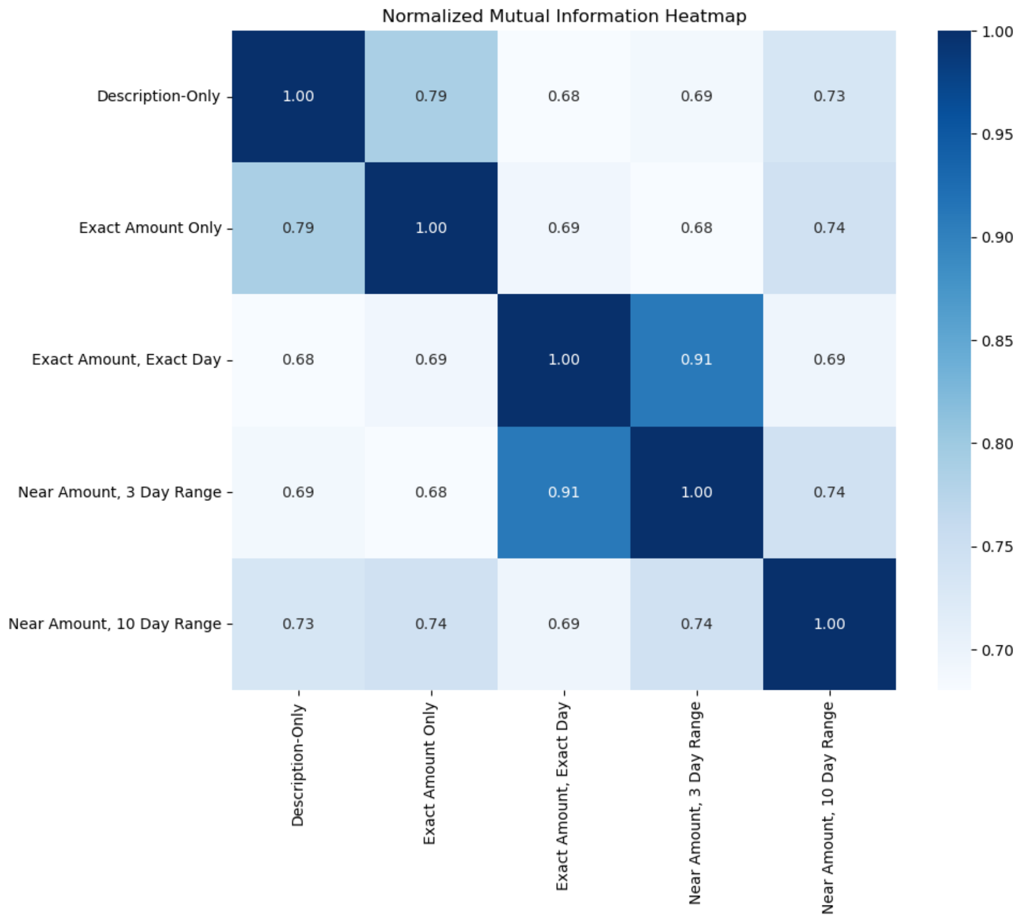
3.2 A Comparison of Payment Grouping Methods with Normalized Mutual Information

Normalized Mutual Information (NMI) allows us to compare the recurring transaction groupings generated by various methods to see how closely their classifications overlap.

First, different groupings were obtained using the distinct methods outlined in the table below:

Method	**Parameterization**
Description Only	Groups together all transactions which have an identical text description. Ignores amount and date.
Exact Amount Only	Groups transactions which have an identical dollar amount. Ignores description.
Exact Amount, Exact Day	Groups transactions which occur on the same day of the month with the same dollar amount. Ignores description.
Near Amount, 3 Day Range	Groups transactions with amounts within \$50 of each other, which also occur with 3 days of the same day of the month.
Near Amount, 10 Day Range	Groups transactions with amounts within \$50 of each other, which also occur with 10 days of the same day of the month.

The last 2 methods listed above utilize the graph-based algorithm. The chart below summarizes the NMI scores for each set of approaches.

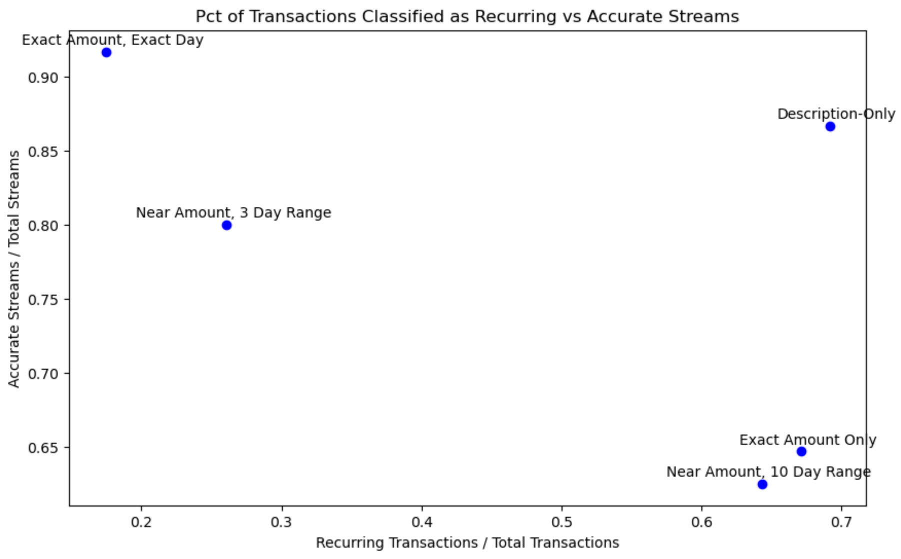


It is promising to see that the custom graph-based methods using transaction-timing and dollar-value proximity are successful in explaining a large share of groupings created by matching transaction descriptions. Because these approaches are orthogonal to each other (the graph algorithm does not use the transaction description), the fact that their NMI score is high provides strong evidence for the accuracy of the graph-based approach.

One limitation of using NMI in this way is that we do not have ground truth labels for comparison. The chart compares approaches to each other, but does not indicate how accurate these approaches are. An interesting extension of this research would be to manually label a sample of recurring payments, so the methods could be compared to known recurring payments.

3.3 The Tradeoff Between Finding More Recurring Payments and False Positives

For each of the methods in the table above, statistics were collected on the how many transactions were classified as recurring, as well as the false positive rate (obtained by manually reviewing output files of the transaction groupings). These statistics are shown below:



With the exception of the Description-Only grouping (discussed below), we can observe an intuitive trend. We find that methods that classify a higher share of payments as recurring tend to have a lower true positive rate for the streams identified. True positives were collected by manually reviewing a spreadsheet output of samples from each method.

In the chart above, the leftmost method does not seem to identify enough recurring transactions. On the other hand, the right-most methods may be classifying too many, since using common descriptions (like “check”) or common amounts (like an even $200), may add too many one-off transactions into a recurring grouping. The graph-based methods (centermost in the x-axis) appear better able to balance the portion of recurring payments.

Although the graph-based methods have inferior true positive rates in the chart above, there a few important considerations not captured in the true positive rate, which tend to favor the graph-based approach. Diving further into the detailed spreadsheet output of the results, the author found that the description-only and amount-only approaches contain a suspiciously high number of missing payments compared to the graph approach. These are likely not true missing payments, but payments which where the description or amount changed slightly. The graph-based approaches tend to capture these better.

4. Identifying Late and Missing Payments

4.1 Description of the Late and Missing Payments Algorithm

To identify late and missing payments, a simple rules-based algorithm is proposed. The algorithm is applied to each grouping of recurring payments individually and is described below:

Late and Missed Payment Algorithm
(applied to each grouping of recurring payments)

Part I: Identify Late Payments

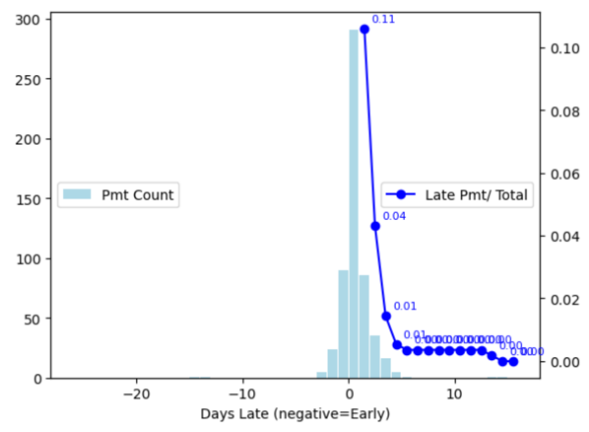
> Step 1: Identify the median day of the month of all recurring payments in the grouping.
> Step 2: Generate a target sequence of evenly-spaced dates of the same length as the recurring payment stream, with each day-of-month replaced by the median day.
> Step 3: Match each date in the original recurring payment stream to its closest date in the target sequence.
> Step 4: Subtract the matching target date from each payment in the recurring stream. Negative values are early payments. Positive values are late.
> Step 5: For a given threshold, classify all payments above that threshold as late.

Part II: Identifying Missed Payments

> Step 6: Identify all dates in the target sequence (generated in Step 2) which were not matched to a recurring payment in Step 3. These unmatched dates are classified as missed payments.

4.2 Percentage of Payments Late by Cutoff Threshold

To evaluate the late payment classification, we can look at the distribution of payments by how many days early/late they are. In the chart below, the light blue histogram shows this distribution.



The distribution is approximately normal with low variance. The vast majority of recurring payments occur on their predicted due date, with the number falling off exponentially with each day that elapses after the predicted due date.

The dark blue line above shows how the percentage of payments classified as late responds to different thresholds for past-due. If we consider 1-day past-due to be late, 11% of the sampled transactions will be classified late. But if a threshold of 3 days is used, less than 1% of payments are considered late.

4.3 Missed Payment Discussion

The missed-payment detection algorithm did surprisingly well on the sample considering its simplicity. When its input was a valid monthly payment sequence, it appeared to function nearly perfectly. On the sample, it discovered 27 true positives and only 1 false positive (excluding cases where the input was not a valid monthly grouping). This translates to approximately 2.5% of sampled payments being classified as missed, a seemingly reasonable estimate.

Note: While reviewing the results of the missed-payment detection algorithm it was discovered that the algorithm is prone to identifying spurious missed payments at the beginning or end of a sequence. Since this can be easily adjusted for by adding a rule to the algorithm, these end-of-cycle errors were disregarded in the analysis above.

Conclusion

In this paper, the author presented two custom algorithms applicable to the problem of classifying transactions in bank statements;

1. A **graph-based algorithm for identifying recurring payments**, which uses transaction-amount similarity and the cyclical pattern of payment dates to connect transactions in a pairwise-manner, then builds a graph in which connected components represent the groupings of recurring payments.
2. A **rules-based algorithm for identifying late and missing payments**, where the median day-of-month is used construct a sequence of target dates and each recurring payment is paired with its nearest target date. Here, the difference between a recurring payment date and its nearest target provides an estimate for how late (early) the transaction is and any unpaired target dates are presumed to be missing payments.

The recurring payment groupings were compared for various identification methods using Normalized Mutual Information and select sampled statistics for their accuracy.

A distribution of late payments was examined to assess how various late thresholds impact the positive classification rate and simple statistics were reported for missing payments.

Several areas for potential future research were also identified. These include:

- Extending the algorithms to apply to non-monthly patterns (e.g. biweekly or quarterly)
- Manually labeling a set of recurring payments and use NMI to see how various methods compare to the ground truth.
- Running all results on a larger sample size.
- Adjusting the recurring payment algorithm to include information about payment description.
- Adjusting the missed-payment algorithm to exclude spurious beginning-of-sequence or end-of-sequence positives.