CLIENT BEHAVIOR ON CREDIT PAYMENTS AT A BANK IN THE EASTERN EUROPE

Aliaksandr Nekrashevich

1. The (somewhat incomplete and simplified) Story

Although one rarely hears about banking in the ancient times, financial industry has a long history and significant impact on the modern society. The concept of exchange begins with increasing difficulties in the individual independent existence and is characterized by the split of responsibilities. Modern history believes that long time ago humans were changing their belongings naturally. One of the first known bank prototypes are assumed to give grain loans to farmers and traders around 2000 BC in Assyria, India and Sumeria. However, with increasing complexity of the world, exchange became complicated. As a result, humans invented an additional concept called money, which allows to remember price for each of the goods instead of the pairwise conversion table. After using grain-money and similar alternatives, there is some historical evidence that money was done from gold, silver or copper, and had itself the proposed value. It has moved to paper and electronic alternatives later.

A very common situation is when someone has too much money, and another individual has an idea or need but not enough money right now. Although modern society can solve this problem by IPOs, shares and derivatives, this project is not focused on them. This situation can also be addressed by issuing a credit or loan. Naturally, the question is how to make the return process reliable, and who should be issued a credit. Moreover, there is another interesting pattern. It has turned out that the approach used by ancient kings of collecting additional money in the case of war by force was not very effective, and therefore the population has often hated their governments. Indeed, it is just paying more when it is not really needed and without getting any benefits. Alternatively, the credit methodology made it reasonable. Occasionally, it has turned out that deposits are a good source for money needed for issuing credits.

To make all these scenarios transparent and effective, there is a branch of economics called banking. Banking has a fairly long history of existence and is now considered one of the most conservative and respected industries. The most standard way banks work is attracting money by issuing deposits and lending money by issuing credits, with obtaining the marginal profit in the interest rates. Although the profit is somehow delayed in time and the nature of money changes rapidly, this simple schema has proven to work over ages. However, this is not the only way banks make their profits. For example, another option is investment banking, which itself can be targeting stock markets, cryptocurrencies or expensive metals. A good introduction into the history and art of banking can be found in [1]. A short reading about the connection between deposits and loans is provided in [2].

1.1. Problem Formulation.

This work addresses the problem of issuing a credit loan for the purpose of purchasing a car. The information was openly presented on a competition the author has opportunity to participate at by one of the Belarusian banks (more precisely, BNB-bank, or Belaruski Narodny Bank) [3]. This bank is specialized on medium-small business operations, industrial and personal loans, and is considered one of the most persistent links between the Republics of Belarus and Georgia. The main personal credit directions of BNB bank are car loans (9 credit lines) and mortgages (4 credit lines).

Date: February 24, 2022.

The competition was called Imaguru Datathon 2019 [4], and was held in Minsk, Belarus by Imaguru Startup Hub [5]. After the dataset is revealed, the teams have 3 days to complete their prototype and present for the jury. Note that it is somewhat a hobby contest, not so much about prizes but more about networking and happy data science. The dataset contains the payment history for several months, which is sometimes incomplete and would be described in one of the next sections. Ideas studied in this project are in part related to what was done at the time of the competition, as well as to the author's industrial experience.

The goal is to understand the client behavior and repayment policies based on their internal features. Although credit scoring is working, clients behave differently. The natural question is then how to use this variability in client behavior to make the bank better? Note that financial atmosphere and economic ecology are very risky in Belarus, which adds additional challenges to the problem.

It is worth to explain how data aspects are addressed in this problem. Typical volume of loans that banks issue varies between financial institutions, and can be huge for the major banks. Velocity is typically not very significant, since every applicant has already passed credit scoring. The story is sometimes different for fast short-term loan businesses, where margins are so huge that companies can risk loosing some money on bad clients. However, since in many post-soviet countries collectors are somewhat criminal and therefore very effective, short-term loaning has a lot of their credits paid back with enormous margins. In any case, the described above bank is not involved in this type of business at the author's knowledge. Variety is represented by the diversity of the client behavior and their payment policies. And veracity also needs to be addressed, since the data is regularly incomplete, especially when the record is provided while the loan is not paid completely at the time of the snapshot.

The value of Data analytics for this particular case is hard to identify explicitly. But by understanding the client behavior, the bank can adjust its credit policies and provide a more reasonable recommendations to their customers. Although client behavior is a very hot topic in banking, there is not much understanding and precise results about it. Usually, the problem is in granularity of behavior patterns and confidence of the findings. This work tries to shed some light and provide some explicit recommendations that are easy to follow and can improve capital stability and bank profitability. Finance is very much into risk and profit. By understanding reliable clients and making products more attractive to good customers, the bank can make its capital less risky. And once capital becomes less risky, it can be re-invested, which is how the bank works and what leads to additional profits.

2. Dataset Description

Dataset consists of two parts with dynamic and static information. Dynamic information contains repayments activities related to the main loan body. Static data contains the initial information about the client and the contract. These two parts are naturally separated into different CSV files. Below is the list with the fields available in the dataset alongside measurements explanation.

Measurements in Static Dataset. Static table has the following 12 columns:

(1) **CONTRACT_ID** – the contract identifier. The contract is a loan for a car purchase.
(2) **CLIENT_ID** – the client identifier.
(3) **CONTRACT_SUM** – credit amount (in a hidden currency, called Datathon Crona). Please note that the this data is normalized, since banks are very rarely good about sharing the exact financial information.
(4) **GENDER** – the gender of the client (M - Male, F - Female).
(5) **CAR_CATEGORY** – the category of the purchased automobile. There are five different categories. The minimal budget category is 1, the maximal premium is 5.
(6) **TERM** – the amount of months for credit repayment.
(7) **AGE** – the age of the client at the time the loan was issued.
(8) **LOAN_TO_INCOME** – the ratio of the credit amount to the monthly client revenue
(9) **PAYMENT_TO_INCOME** – the ratio of monthly client payment to the monthly client revenue
(10) **DOWNPAYMENT** – the ratio of client self-participation in the car purchase. DOWNPAYMENT = 1 - (CONTRACT_SUM / cost of the automobile)
(11) **GRACE_PERIOD** – the length of the grace period in months. At the beginning of the contract during this period, the interest rate is lower than the regular one afterwards. If GRACE_PERIOD = 0, there is no period with discounted interest rate.
(12) **RATE_CHANGE_AFTER_GRACE** – how the interest rate changes after the grace period is over.

Measurements in Dynamic Dataset. Dynamic table has the following 4 columns:

(1) **CONTRACT_ID** – contract identifier. The contract is a loan for a car purchase. This key links static and dynamic tables.
(2) **PERIOD_ID** - the month after the contract was issued. When PERIOD_ID = 1, it is the first month after the loan was provided.
(3) **REPAYMENT_SCHEDULED** - amount of payment at the current period according to the contract.
(4) **REPAYMENT_ACTUAL** - factual payment by the client in the current period. When NULL, it means this period has not yet arrived.

There are 3792 contracts in total, among them only 91 are completed at the time the dataset was provided, which means that 3701 other contracts are provided with missing values which are yet to happen later. Figure 1 compares amounts of completed and in-progress contracts. The maximal length of a completed contract is 18.

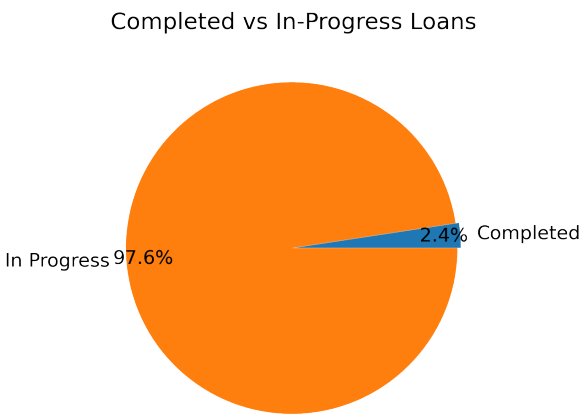

FIGURE 1. Completed and in-progress contracts.

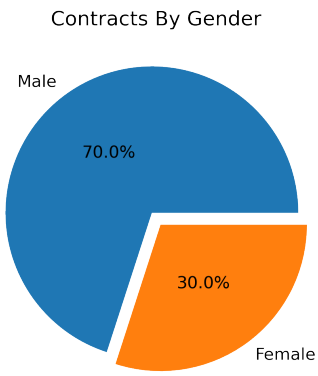

FIGURE 2. Client genders in the dataset.

Gender-related algorithms are acceptable in Belarus, therefore Figure 2 compares amounts of contracts issued to males and females. There are 2655 male contracts and 1137 female.

Majority of the contracts are buying car category 2, as depicted at the Figure 3, with a very few purchases of the most expensive cars. The most popular loan term is 120 months or 5 years, as shown in the Figure 4. Overall pair-plot is presented in the Figure 5.

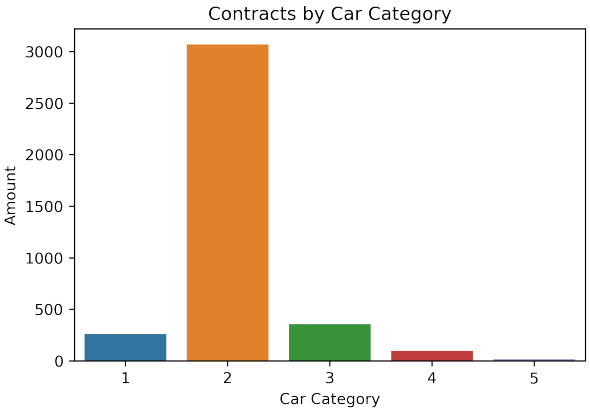

FIGURE 3. Contracts by car category.

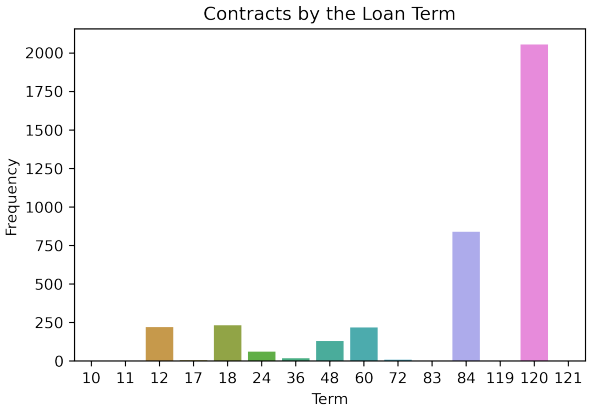

FIGURE 4. Contracts by credit duration.

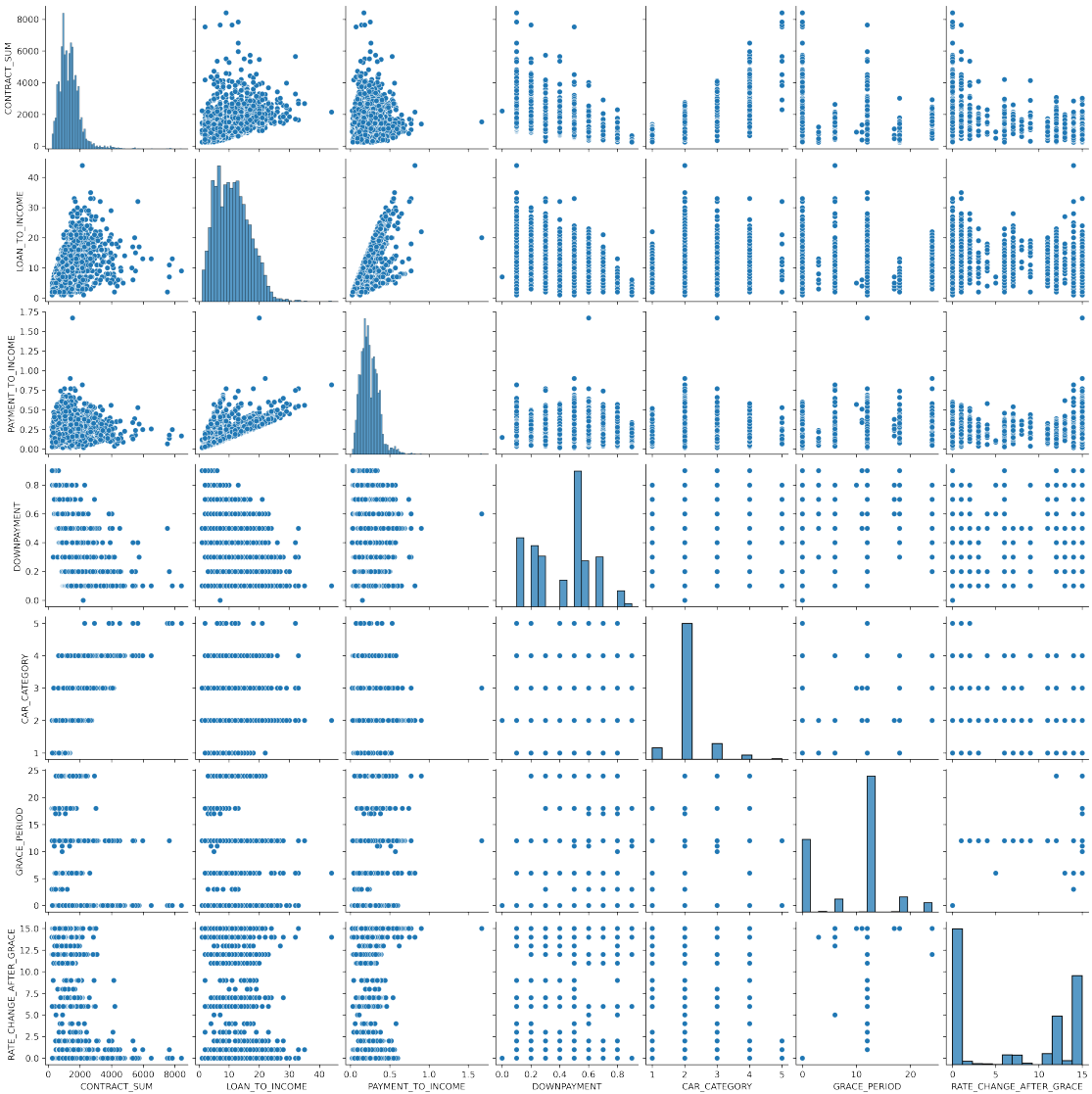

FIGURE 5. Pairwise interaction of static features.

3. Approach Overview

This work investigates two main directions that can be useful for bank operations. The first direction is related to time series clustering, and the second direction is about client reliability. Visual structure of the project is presented at Figure 6.

For time series clustering, begin by clustering first 12 months of each contract when these 12 months do not have missing values. There is already an issue about it, since only small part of time series is at least 12 months long. This clustering is done first with hierarchical approaches to guess the number of clusters, and then fine-tuned with approaches that fix cluster number in advance. After clusters are obtained, the next question is how to assign clusters to all other time-series. This is done by first completing time series, and then applying already learned cluster prediction algorithm. When filling time series, as a side effect one can study which factors influence the next period payment most.

Client reliability studies start by defining reliable clients. After assigning labels according to this definition, it turns out that the created dataset for classification is imbalanced. Therefore, balance-aware algorithms are used for estimating reliability. As a side effect, this project also studies how features influence reliability.

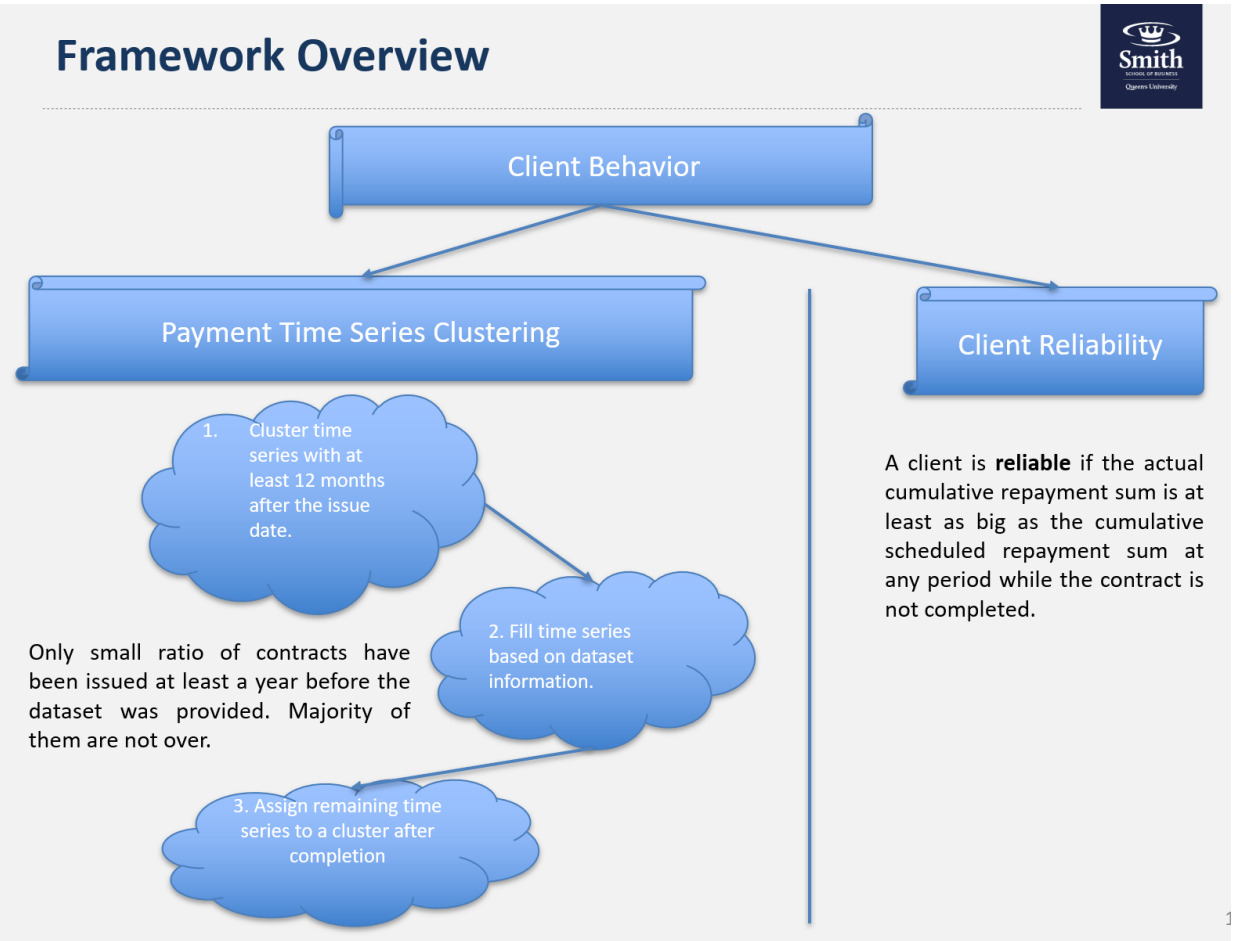

Figure 6. Structure of the project.

4. Time Series Clustering

Proceed in the following steps. First guess the number of clusters on random 20 time series with hierarchical clustering. Then learn algorithms with fixed number of clusters. After that, fit regressors to complete time series. After that is done, one can complete time series and assign clusters to the completed time series.

4.1. **Guessing the number of clusters.** This step is performed by calling several routines from DTAIdistance library, [6]. The selected methods were LinkageTree and HierarchicalTree, and their result is depicted in the graphs below at Figures 7 and 8.

Since the amounts of contracts (and therefore repayment amounts) are somehow different in scale, it is reasonable to normalize repayments over the contract sum. Therefore, all elements in the studied time series are rescaled between 0 and 100 percent, which represents the repayment fraction of the loan size.

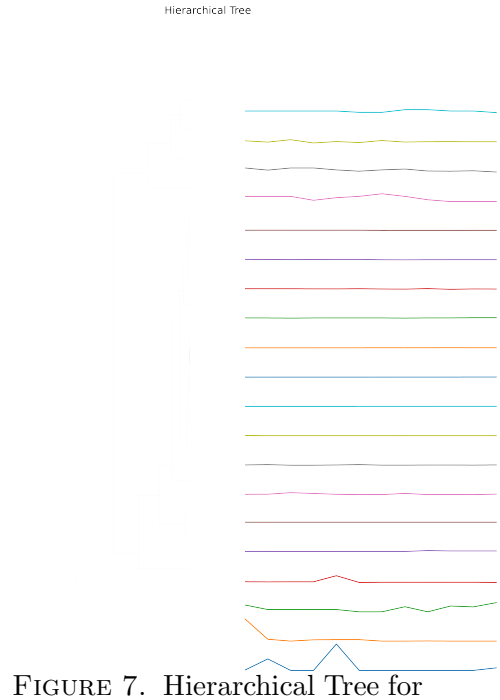

FIGURE 7. Hierarchical Tree for guessing the number of clusters.

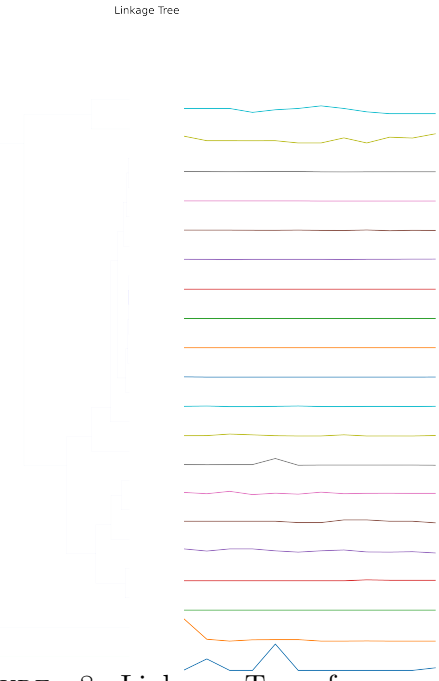

FIGURE 8. Linkage Tree for guessing the number of clusters.

As the figures show, the guessed number of clusters is somewhere from 3 to 5. They can be either flat repayments, or single spike, or double spike, or big initially then flat, or flat and big in the end.

4.2. **Fixed Number of Clusters.** Now fit the algorithms for fixed number of clusters and tune their performance. The most popular Pythonic library for this purpose is [7]. The majority of working approaches are based on DTW-distance, which is described in [8], and a soft-loss version of which is described in paper [9]. Tslearn library [7] has also kernel-based implementation of kMeans. Another important method is kShape, which was first introduced in [10]. There is also kMedoids algorithm based on DTW, available at [11].

It has turned out that DTW-based approach was the most reasonable in this case, with its output present in Figures 9, 10. After trying different cluster sizes, it turns out that having 4 of them makes the most sense. The first cluster represents flat and stable repayments, as scheduled or with not much deviation from it. The second cluster is flat initially, then has a spike covering almost all the loan, and is either zero or flat until the end of the year. The third cluster is flat until the end, with a sudden spike covering full loan size in the end. The fourth cluster is high at the beginning, and is then moderately flat in the end. Note that the scales of the first and the second clusters are different (40 vs 100), therefore what seems somewhat like a spike at the first cluster would be moderately flat at the scale of the second cluster.

Now think why this result makes sense. At a first glance, it looks like the first cluster has much more entries than other clusters. And this is good, since it means that the majority of the clients are stable and reliable. Other clusters look more fancy in terms of exploration

and presentation, but from the point of bank, they are risky, so it looks good when they are smaller compared to the main group.

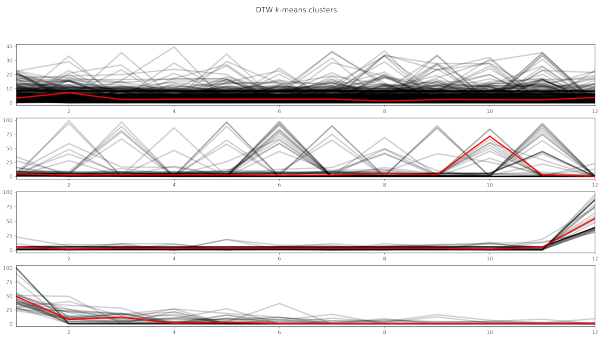

FIGURE 9. DTW-based kMeans clusterization into 4 clusters.

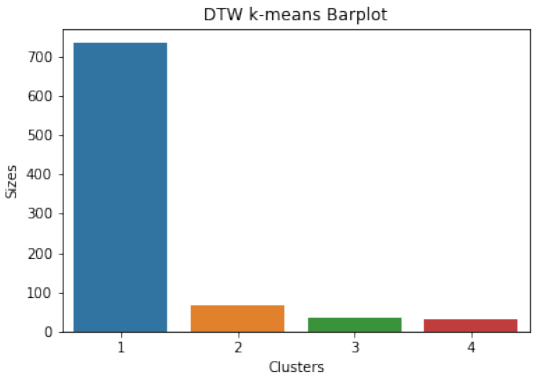

FIGURE 10. DTW-based kMeans cluster sizes.

For a complete picture, it is reasonable to present the results obtained by the other clustering algorithms. However, they typically find only two clusters (flat repayments and single spike), as Figures 11, 13 and 12 show. Therefore, the approach with DTW looks more reasonable.

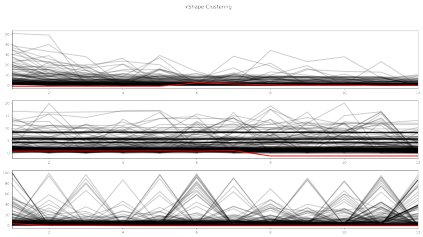

FIGURE 11. kShape clusters.

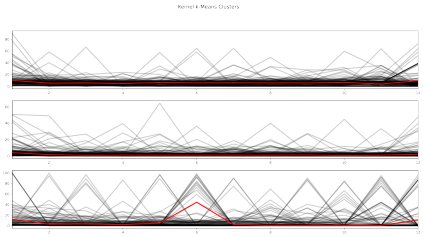

FIGURE 12. Kernel k-Means clusters.

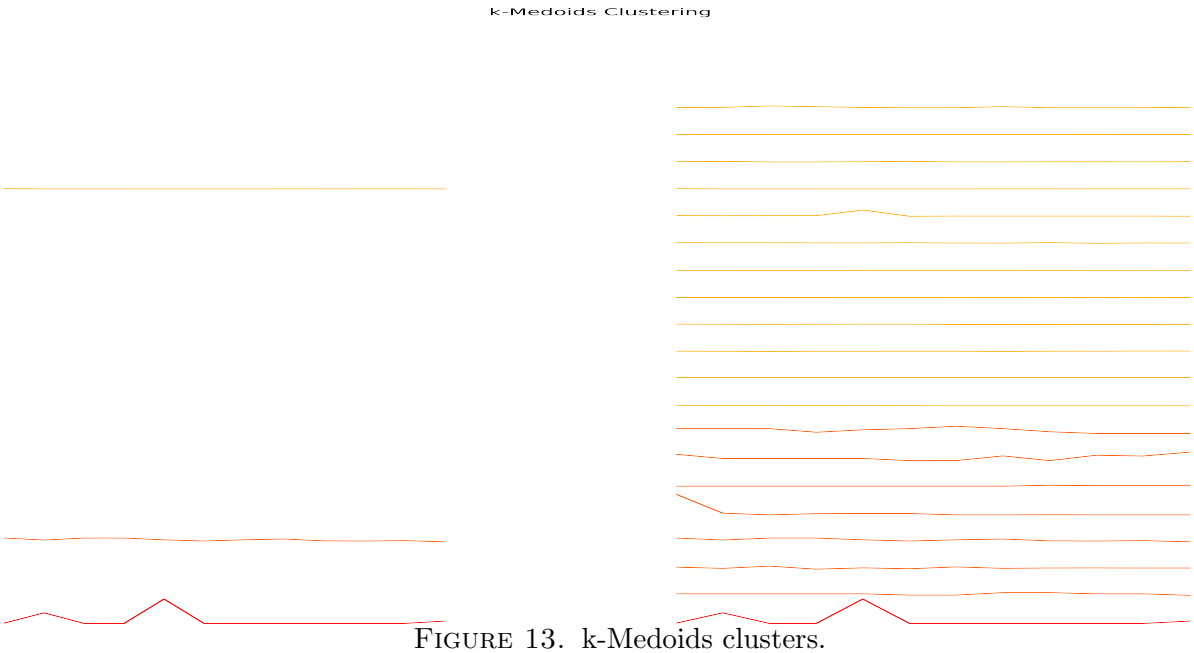

FIGURE 13. k-Medoids clusters.

4.3. **Predicting Next Month.** Due to a significant static context, cluster structure and aggregated time series information, it is unlikely that the regular time series completion

methods would work. Therefore, this project tries to apply general regression models with some historical context as additional features. To explain the result, one first need to describe the set of engineered features.

4.3.1. *Feature Engineering.* The following groups of features were engineered to improve the regression performance:

(1) **CUMSUM_***: Cumulative repayments (actual and scheduled).
(2) **AVERAGE_***: Average repayment until the current period (actual and scheduled).
(3) **RELATIVE_*, RELATIVE_CUMSUM_***: Relative repayments, periodic, average and cumulative, actual and scheduled.
(4) **LOG_*, LOG_RELATIVE_*, LOG_RELATIVE_CUMSUM_*, etc.**: Log-transformation of features (static, cumulative, average, and relative), by using numpy.log1p.
(5) **HISTORY_*, HISTORY_LOG_*, HISTORY_LOG_RELATIVE_*, etc.**: Information from the two previous periods, replaced by scheduled amounts if it is one of the first two periods.
(6) **IS_GRACE_CONSTANT_***: Dummy which shows if payments during the grace were constant.
(7) **RATIO_***: Dummy demonstrating if LOAN_TO_INCOME is above some threshold among (5, 10, 20, 30, 40).
(8) **IS_CAR_***: Indicator dummy on the car category.
(9) **IS_PERIOD_***: Dummy for the first three periods 1-2-3.
(10) **IS_GRACE_ON_***: Indicator if the grace period is on at the time of repayment.

Moreover, it turned out that it might be reasonable to transform target before and after prediction. In this work, the following transformation were tried:

(1) Identical. No transformation applied.
(2) Relative. The target value is divided by the contract sum and multiplied by 100. After the estimation is done, this operation is reversed.
(3) Logarithmic. First logarithm function is applied to the target. After the estimation is done, this operation is reversed.
(4) Log-Relative. The target is first converted to relative percent repayment of the loan sum, and then a logarithm is taken. After the estimation is done, these two transformations are reversed.

4.3.2. *Algorithm Comparison.* To select the best predictor, multiple algorithms from scikit-learn [12] were tried. They are Ridge Regression, Lasso Regression, Huber Regression, Bayesian Regression, Random Forest, Neural Network, Support Vector Regression, and K-Nearest Neighbors. Moreover, boosting algorithms from library XGBoost were tried [13]. These algorithms were trained on all engineered features and all possible target transformations. Baseline was implemented by simply returning the scheduled repayment value. Two evaluation tables were created: Table 1 for predicting relative target, and Table 2 is for predicting precise target. It turned out that Random Forest for log-relative target shows the best performance.

4.3.3. *Explaining Next Month Repayments.* First plot lasso linear regression coefficients of the major impact, which is done in the Figure 14. Bigger payment to income makes payment less, and car category alongside with repayment in the last period make the next payment bigger. Yet another natural driver increasing actual repayment is car category, which is fine since more expensive cars have bigger loans (though usually lower interest rates and longer periods).

	Algorithm	Target Form	MAE	RMSE	MAXERR	EXPVAR
0	Baseline	Id Exact	2.31	7.71	99.17	0.03
1	Ridge Regression	Id Exact	2.45	7.08	99.30	0.13
2	Ridge Regression	Log Exact	1.95	7.28	98.96	0.11
3	Ridge Regression	Id Relative	2.48	7.08	99.39	0.13
4	Ridge Regression	Log Relative	1.91	7.20	99.02	0.12
5	Lasso Regression	Id Exact	2.67	7.19	98.31	0.11
6	Lasso Regression	Log Exact	2.81	7.78	99.04	0.03
7	Lasso Regression	Id Relative	3.43	7.50	96.55	0.03
8	Lasso Regression	Log Relative	3.02	7.80	98.13	0.00
9	Bayesian Regression	Id Exact	2.46	7.08	99.30	0.13
10	Bayesian Regression	Log Exact	1.93	7.27	98.96	0.11
11	Bayesian Regression	Id Relative	2.48	7.08	99.41	0.13
12	Bayesian Regression	Log Relative	1.90	7.19	99.02	0.13
13	Random Forest	Id Exact	2.26	7.08	98.26	0.13
14	Random Forest	Log Exact	1.76	7.19	98.87	0.13
15	Random Forest	Id Relative	2.27	7.06	98.53	0.14
16	Random Forest	Log Relative	1.73	7.12	98.81	0.14
17	Xgboost	Id Exact	2.64	8.12	99.01	-0.11
18	Xgboost	Log Exact	2.94	7.95	99.45	0.05
19	Xgboost	Id Relative	2.28	7.57	98.56	0.02
20	Xgboost	Log Relative	2.35	7.53	98.75	0.09
21	Sklearn Neural Net	Id Exact	2.52	7.04	98.60	0.14
22	Sklearn Neural Net	Log Exact	2.21	7.21	98.57	0.10
23	Sklearn Neural Net	Id Relative	2.26	7.14	99.70	0.13
24	Sklearn Neural Net	Log Relative	3.51	10.53	176.56	-0.92
25	Svr	Id Exact	4.85	9.09	102.32	-0.04
26	Svr	Log Exact	3.27	8.09	100.00	0.05
27	Svr	Id Relative	5.30	8.90	104.36	0.10
28	Svr	Log Relative	3.83	8.35	100.53	0.03
29	Knn Regressor	Id Exact	2.58	7.41	98.80	0.05
30	Knn Regressor	Log Exact	1.99	7.25	98.84	0.11
31	Knn Regressor	Id Relative	2.48	7.33	98.88	0.07
32	Knn Regressor	Log Relative	1.94	7.20	98.89	0.12

TABLE 1. Percent Prediction Metrics

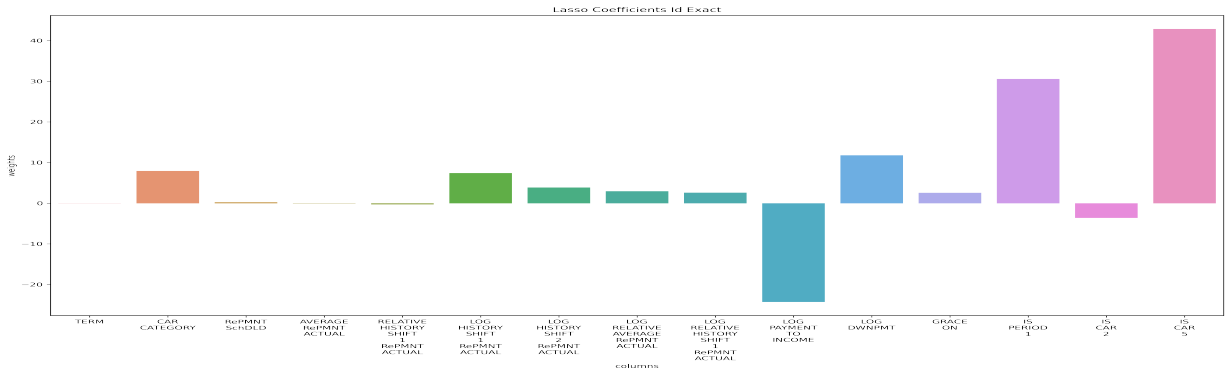

FIGURE 14. Lasso Coefficients for the Untransformed Repayment Target.

Now study Ridge coefficients for relative repayment scale in Figure 15. In the relative scale, the main positive drivers are historical payments, down-payment and the ratio of payment to income, and the main negative drivers are age and understanding that grace is constant.

	Algorithm	Target Form	MAE	RMSE	MAXERR	EXPVAR
0	Baseline	Id Exact	26.38	98.95	1947.34	0.01
1	Ridge Regression	Id Exact	29.40	93.35	1867.76	0.07
2	Ridge Regression	Log Exact	22.94	95.14	1897.17	0.07
3	Ridge Regression	Id Relative	29.99	93.19	1840.46	0.08
4	Ridge Regression	Log Relative	22.57	94.12	1880.58	0.08
5	Lasso Regression	Id Exact	31.33	94.10	1884.10	0.06
6	Lasso Regression	Log Exact	31.86	100.03	1948.60	-0.00
7	Lasso Regression	Id Relative	44.18	98.77	1865.04	-0.03
8	Lasso Regression	Log Relative	34.79	98.84	1916.25	-0.01
9	Bayesian Regression	Id Exact	29.41	93.34	1867.65	0.07
10	Bayesian Regression	Log Exact	22.87	95.09	1896.49	0.07
11	Bayesian Regression	Id Relative	30.05	93.20	1839.58	0.08
12	Bayesian Regression	Log Relative	22.50	94.07	1879.97	0.08
13	Random Forest	Id Exact	26.89	93.30	1836.53	0.08
14	Random Forest	Log Exact	21.06	94.69	1882.06	0.07
15	Random Forest	Id Relative	27.92	93.60	1842.35	0.07
16	Random Forest	Log Relative	20.73	93.75	1874.31	0.08
17	Xgboost	Id Exact	31.06	103.31	1906.52	-0.11
18	Xgboost	Log Exact	33.40	101.29	1945.64	0.02
19	Xgboost	Id Relative	26.38	96.50	1894.07	0.02
20	Xgboost	Log Relative	26.44	96.89	1898.13	0.05
21	Sklearn Neural Net	Id Exact	30.68	93.15	1869.03	0.08
22	Sklearn Neural Net	Log Exact	26.61	94.64	1878.67	0.05
23	Sklearn Neural Net	Id Relative	28.83	96.26	1909.72	0.03
24	Sklearn Neural Net	Log Relative	43.56	142.61	2384.55	-1.16
25	Svr	Id Exact	51.88	107.29	1952.76	0.05
26	Svr	Log Exact	37.49	102.89	1946.62	0.02
27	Svr	Id Relative	70.69	121.10	2003.99	-0.04
28	Svr	Log Relative	43.24	103.85	1905.20	0.04
29	Knn Regressor	Id Exact	30.28	96.35	1878.21	0.01
30	Knn Regressor	Log Exact	23.62	95.36	1903.72	0.06
31	Knn Regressor	Id Relative	29.78	96.18	1862.24	0.02
32	Knn Regressor	Log Relative	23.06	94.42	1878.84	0.07

TABLE 2. Value Prediction Metrics

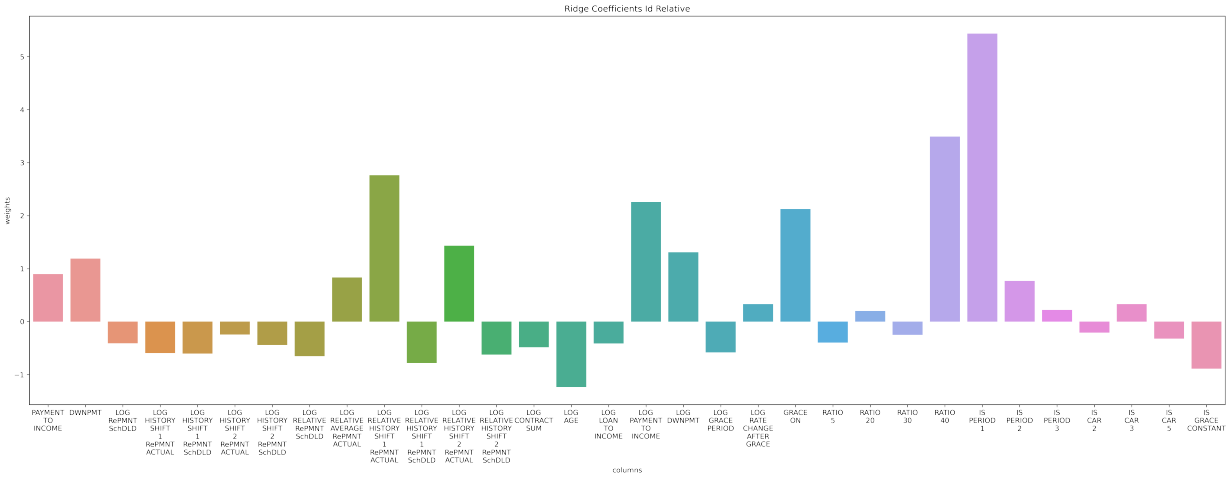

FIGURE 15. Ridge Coefficients for the Relative Repayment Target.

Tree-based algorithms are not easily demonstrate the sign of impact, however they can list the most important features. For the log-relative target in XGBoost, Figure 16, the most important features are contract sum, age, relative scheduled repayment, 1-step history, loan to income, relative cumulative repayment, payment to income, downpayment and term. For

the actual value target in XGBoost, Figure 17, the contract sum, actual scheduled payment, age, average repayment and 1-step history are the most important.

Random forest also can provide feature importance, which are shown in Figures 18 and 19. For regressing the exact repayment, the most relevant columns are history, log of cumulative repayment, average repayment and scheduled repayment. For predicting the relative repayments, the most important features are history, cumulative sum, down-payment and contract sum.

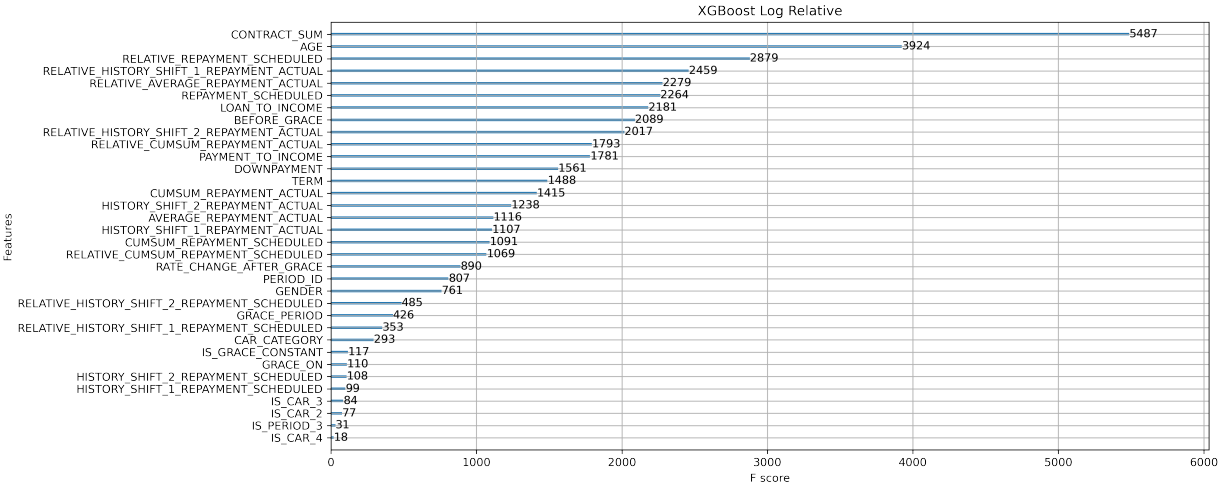

FIGURE 16. XGBoost Feature Importance for the Relative Repayment Target.

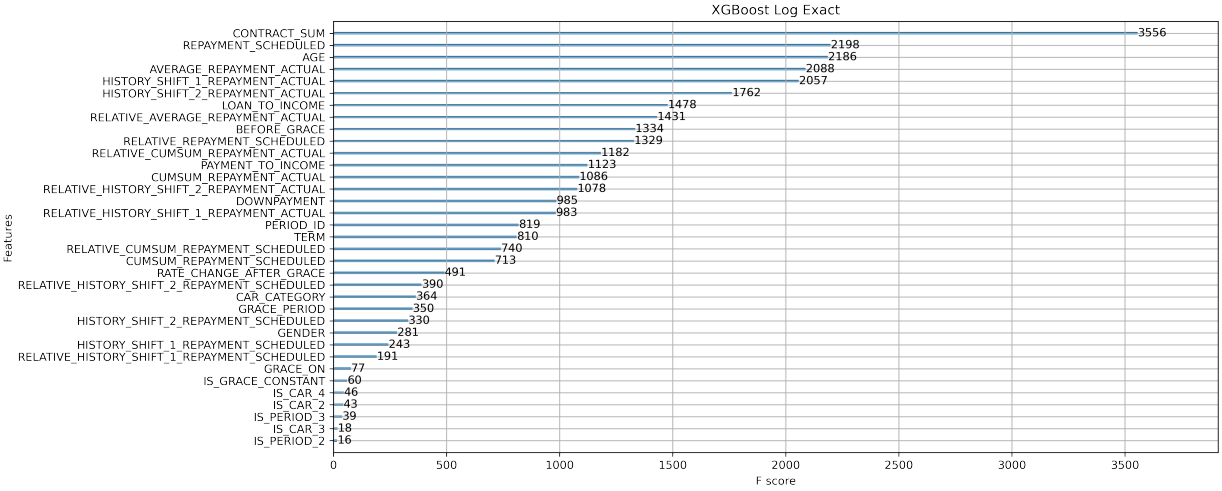

FIGURE 17. XGBoost Feature Importance for the Untransformed Repayment Target.

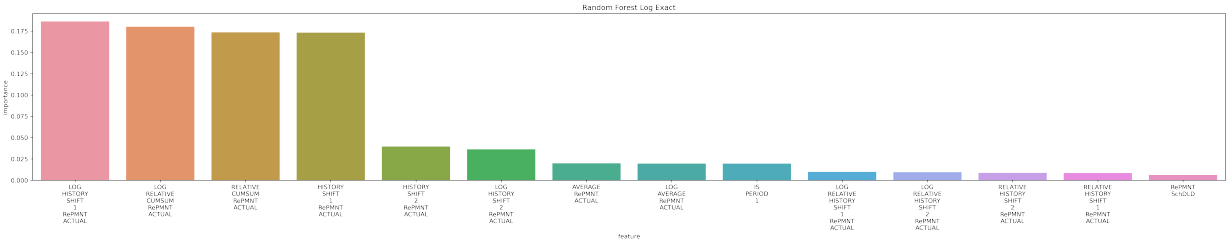

FIGURE 18. Random Forest Feature Importance for the Log Exact Repayment Target.

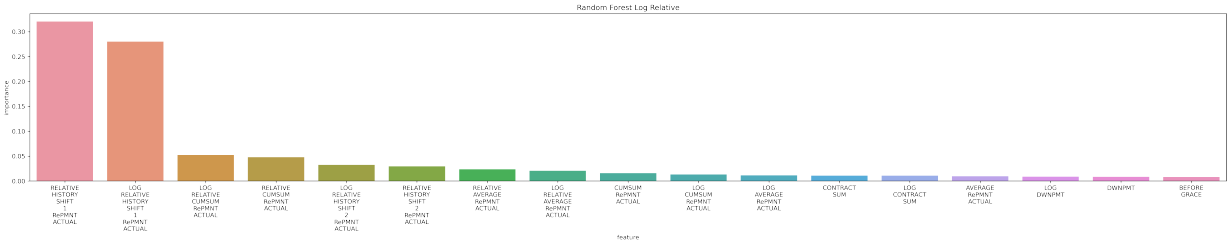

FIGURE 19. Random Forest Feature Importance for the Log Relative Repayment Target.

4.4. **Applying algorithms.** After applying classifier, the sizes of the obtained clusters are depicted in the Figure 20. The main cluster is still dominating, however sizes of other clusters have slightly increased.

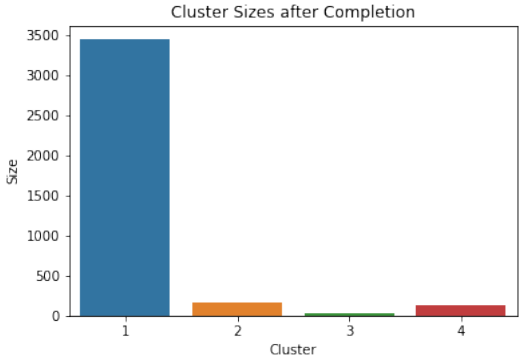

FIGURE 20. Cluster Sizes as the result of applying prediction framework.

The examples of time series completion are provided in the following figures. Although it is not ideal and is yet far from perfect, it works reasonably in multiple cases at Figures 21 and 22. Note that predicting the exact values is not very important, it is more about the shape of the curves. And it is already good if it predicts increase/decrease in the majority of points.

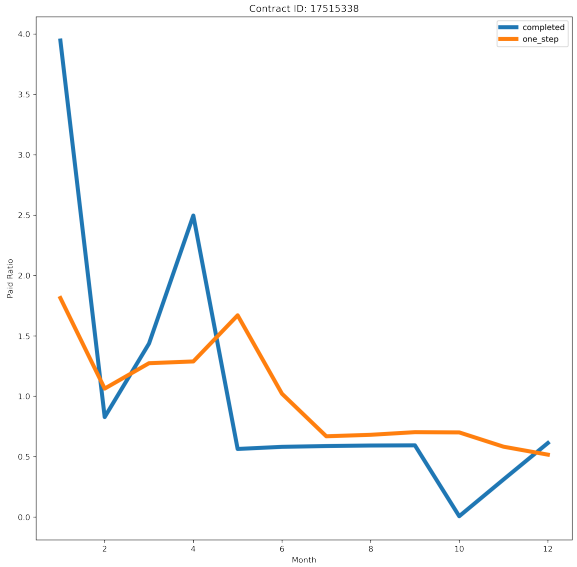

FIGURE 21. Completed Time Series

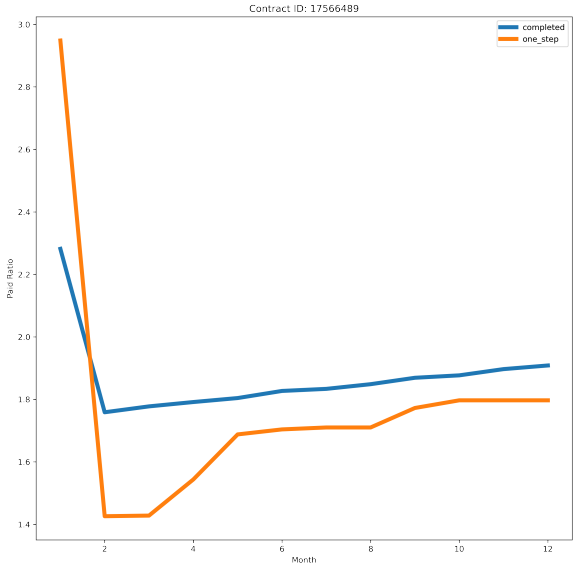

FIGURE 22. Completed Time Series

5. Reliability

Begin by defining reliability. It somehow linked to our dataset and problem setting, and therefore is just an assumption and does not pretend to be a general truth. Reliability is important, since reliable money can-be reinvested and improve Funds Transfer Pricing. Reliable clients are less risky, though still can be sometimes less profitable in terms of interest rate (if they close the learn too fast). However, by modeling interest-rate income and reliability on loan repayment separately, these two patterns can be mixed together and then used for balancing risk and profit.

Definition 1. *A client is* **reliable** *if the actual cumulative repayment sum is at least as big as the cumulative scheduled repayment sum at any period while the contract is not completed.*

By applying this definition, one can get the 0-1 labels that tell if the clients are reliable. The training set has 3348 reliable and 444 unreliable clients, which makes it an imbalanced classification problem. Training imbalanced Logistic Regression and imbalanced Random Forest results with the following performance table:

	algorithm	accuracy	precision	recall	f1-score	rel. prec.	rel. rec.	unrel. prec.	unrel. rec.
0	Logistic Regression	0.58	0.82	0.58	0.66	0.91	0.59	0.16	0.16
1	Random Forest	0.80	0.81	0.80	0.81	0.89	0.88	0.21	0.21

Table 3. Reliability Prediction

There is a way to study Lasso coefficients explicitly, as depicted in Figure 23. It turns out that term, car category and age improve reliability, as well as rate change after grace (the latter may be due to the loan condition design and therefore is not so important). Reliability decreases with loan-to-income ratio, grace period and payment-to-income ratio, as well as with the contract sum.

Random Forest classifier performs with better accuracy, but only shows importance of the features, as shown in the Figure 24.

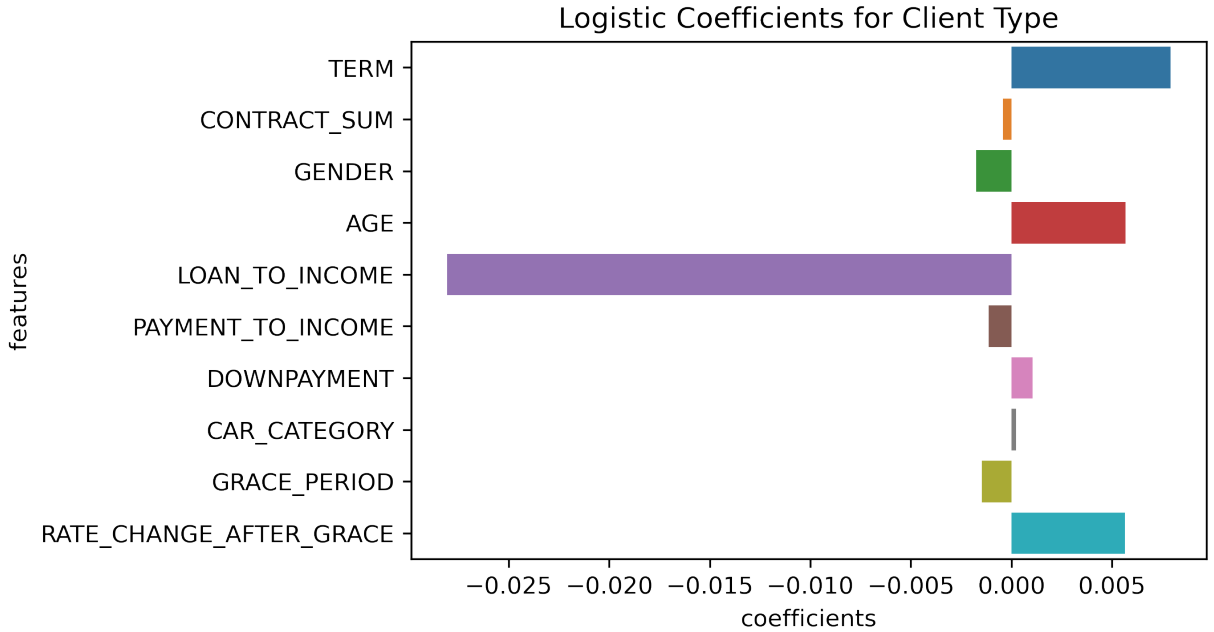

Figure 23. Reliability with Logistic Regression

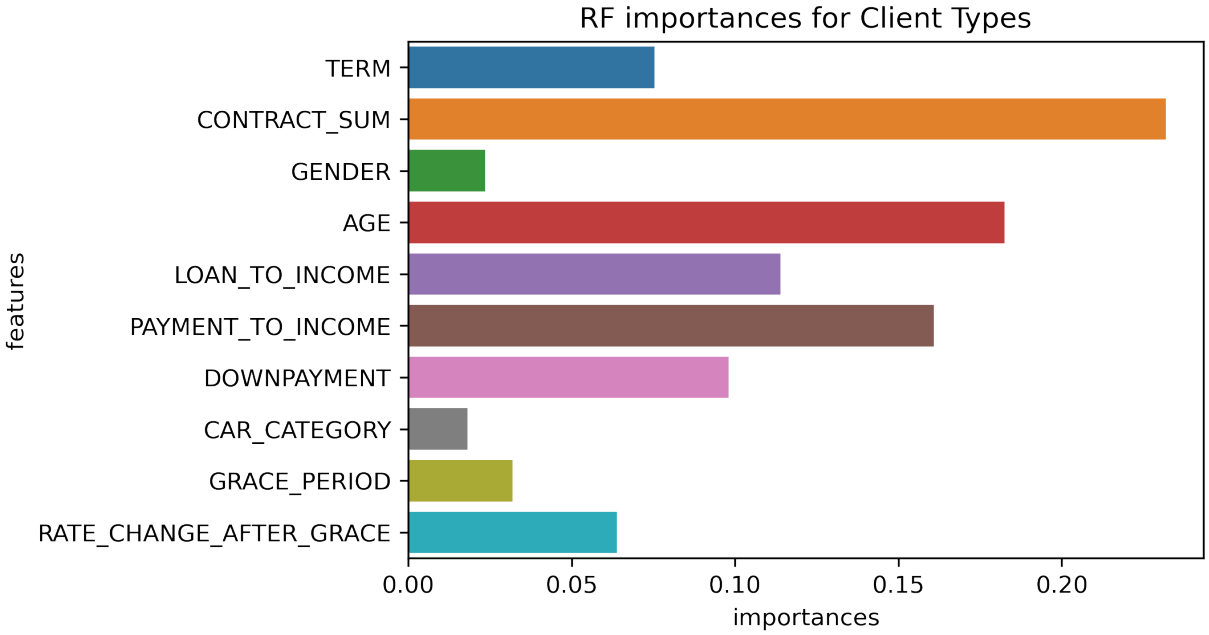

FIGURE 24. Reliability with Random Forest

6. Conclusion

This work contains client repayment clustering into four behavior patterns. There is also interpretation of the next-period repayment. As another direction,this work studied the client reliability in the terms of forecasting and interpretation.

7. Acknowledgements

Author is grateful to some of his Datathon 2019 teammates which behaved effectively and productively during the competition. More precisely, it was only Artsem Shekh (`https://www.linkedin.com/in/prestal/`).

References

[1] Moorad Choudhry. *An introduction to banking: liquidity risk and asset-liability management*. John Wiley & Sons, 2011.
[2] Nasr Georges. *Fund Transer Pricing – From A Standardized To An Advanced Approach*. Amazon Kindle Store, 2017.
[3] Official web-page of BNB-bank: www.bnb.by.
[4] Imaguru Datathon 2019 Web-Page: https://indatalabs.com/blog/imaguru-event-overview.
[5] Imaguru Startup Hub Web-Page: https://imaguru.by.
[6] Wannes Meert, Kilian Hendrickx, and Toon Van Craenendonck. Dtaidistance: Library for time series distances.
[7] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.
[8] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3):678–693, 2011.
[9] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International Conference on Machine Learning*, pages 894–903. PMLR, 2017.

[10] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870, 2015.

[11] Andrei Novikov. Pyclustering: Data mining library. *Journal of Open Source Software*, 4(36):1230, apr 2019.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.