Customer Retention through an API-Driven Survival Analysis within a Microservice Architecture For "Global Credit" UCO CJSC Spring 2024

Anna Manasyan
BS in Data Science
American University of Armenia

Karen Hovhannisyan
American University of Armenia

***Abstract*—Customer retention is crucial for business success, ensuring a steady revenue stream and fostering brand loyalty. Modern companies invest significant resources in customer retention efforts, including improved service, loyalty programs, and personalized experiences. This paper presents a comprehensive approach to enhancing customer loyalty through predictive modeling and data pipelining for "Global Credit" UCO CJSC, focusing on their CashMe loan product. The project employs survival analysis (the Accelerated Failure Time (AFT) technique) to predict the likelihood of customers returning for another loan, complemented by a data pipeline spanning database setup, modeling, and an API for operational integration. Leveraging Docker for containerization, PostgreSQL for database management, and FastAPI for API development, the project streamlines workflow and facilitates seamless interaction between systems.**

***Index Terms*—loan company, retention, survival analysis, pipeline**

I. Introduction

Customer retention is an essential component for the sustained success of any business. Beyond the initial sale, retaining customers ensures a steady revenue stream and promotes brand loyalty. Thus, most modern companies allocate a great deal of time and resources to customer retention efforts. Some common practices include improved customer service, loyalty programs, and personalized experiences.

"Global Credit" UCO CJSC (Company) is one of the leading universal credit organizations in the Republic of Armenia, offering its customers a wide range of loan services spanning various sectors, including consumer, business, mortgage, and agriculture. The Company stakeholders expressed an interest in enhancing customer retention, specifically for one of their consumer loan products - the CashMe loan. This online loan option is available to individuals aged 21-65, offering a maximum sum of 900,000 AMD with a repayment period of up to 18 months.

This project entails the development of an end-to-end product, beginning with the definition of the business problem and culminating in the delivery of a **software solution** that will help the Company make data-driven decisions and improve its customer retention efforts. To elaborate, this project treats the business problem as a time-to-event modeling issue and leverages the power of survival analysis to predict the likelihood of a specific customer returning to apply for another loan in the Company. Additionally, in order to mirror real-world business operations, the project entails the development of a data pipeline. The pipeline starts off with the initiation of a PostgreSQL database, extends to modeling, and culminates in the creation of an API. The API is intended to make the modeling results accessible to the customer service operators, enabling them to identify potential returning customers, target specific clients, and document the strategies used in their retention efforts.

II. Problem Definition

As mentioned earlier, the first step of the project was to define the business problem. It is important to note that throughout the execution of the entire project, regular meetings were held with the stakeholders of the Company in order to be in line with their requirements, keep them updated on progress, and get their expert opinion. Upon these discussions, the business challenge was identified as stimulating CashMe clients to promptly reactivate another CashMe loan following the closure of their previous one. Thus, the problem this project aims to solve was established to be **estimating the conditional probability of a customer returning to take another CashMe loan, given that the client has closed his/her latest loan.** The Company can utilize this information to prompt customers to apply for another CashMe loan shortly after closing their previous one.

III. Research & Methods

A. Data Collection

With the business problem clearly defined, the next natural step was exploring the Company's database. The Company database is rather extensive, comprising vast amounts of information sourced both internally and externally. In close collaboration with field experts, we extracted a set of potentially impactful features for this project, which mainly revolve around customer characteristics and behavioral patterns. In the

end, a total of seven tables were extracted (***Table 1***). A detailed description of the composition of each of the tables can be seen in Appendix 1. It's important to know that the extracted dataset includes information on CashMe loans distributed throughout the year 2021.

TABLE I
ORIGINAL DATASET

Table Name	Description
marz	Information regions in Armenia.
consumer_client	Details about clients.
consumer_main	Information about loan applications.
consumer_hc	Information related to disbursed loans.
consumer_family_relation	Family members associated with clients.
eceng_ces_data	Information about the clients' legal proceedings.
eceng_vehicle_info	Information about the vehicles of the client.

B. Architecture

The project consists of 3 main parts:

- Database
- Modelling
- API

To streamline and standardize the workflow, the Docker platform was utilized for this project. Docker is a containerization platform that enables developers to package applications and their dependencies into standardized units called containers. Containers are isolated and include everything required for running the application, eliminating reliance on the host system and ensuring seamless functionality [2]. This project entailed the development of multiple containers; thus, the Docker compose tool was utilized to define and run a multi-container application. The PostgreSQL relational database management system image [3] was chosen to store data, and Admin for PostgreSQL Management image was used to manage the database [1]. For database setup and operation, Python was used, employing key packages such as SQLAlchemy in collaboration with PostgreSQL database adapter psycopg2.

C. Modelling

The next important step of the project was choosing the modeling technique that would be implemented. Given the problem definition, survival analysis emerges as a fitting statistical tool for estimation and prediction, offering a technique to analyze the time until the occurrence of an event of interest, in this case, the reactivation of a CashMe loan. Before discussing this further, let's define a few key terms:

- **Survival Time:** The time until the occurrence of the event of interest. In our case, it's the time between closing a CashMe loan and reactivating the next.
- **Survival Function**: This function, typically denoted as $S(t)$, gives the probability that a subject survives beyond time t. It represents the probability that the event of interest (loan reactivation) has not occurred by time t.
- **Hazard Function:** The hazard function, denoted as $h(t)$, represents the instantaneous rate of occurrence of the event of interest at time t, given that the subject has survived up to time t. It provides insights into the risk of experiencing the event at a particular time.
- **Cumulative Hazard:** The cumulative hazard function $H(t)$ provides information on the total accumulated risk of experiencing the event of interest that has been gained by progressing to time t.

There are three distinct groups of survival models:

- **Nonparametric models**, such as the Kaplan-Meier Estimate, do not assume any parametric form for the survival function or the hazard function.
- **Parametric models**, such as the Accelerated Failure Time (AFT) Model, assume that survival time follows a known distribution. The Weibull, the log-logistic, and the log-normal distributions are commonly used for survival time.
- **Semi-parametric models**, such as the Cox proportional hazards model, do not require an assumption about the shape of the hazard function; however, they make assumptions about the effect of covariates on the hazard function.

Initially, Kaplan-Meier was used to analyze the survival curve and get insights into the overall survival distribution. However, as it does not consider the effects of exploratory variables, the choice for a main modeling technique was between the semi-parametric and parametric models. In the early stages of the project, it became apparent the proportional hazards assumption required for the Cox proportional-hazards model failed for most exploratory variables. This indicates that the effects of most variables on the hazard rate varied over time. Therefore, a strategic decision was made to implement the modeling using the parametric Accelerated Failure Time (AFT) model [5]. .

D. API

APIs, or Application Programming Interfaces, serve as intermediaries, allowing different software systems or platforms to interact with each other. Among the various types of API methods, the GET method is designed for retrieving data or information from a specific resource such as a database. On the other hand, the POST method is employed for submitting data to a designated resource, often used when there's a need to create or update data on the server. These two types of requests were integrated into the project [6]. FastAPI - famous for its efficiency and ease of use, was chosen for the API development process [4]. Data interactions were administered through SQLAlchemy, ensuring smooth access and manipulation within the PostgreSQL database. The API testing was conducted using the ASGI server Uvicorn. This methodology helped the develop of a robust, efficient, and user-friendly API capable of seamlessly interacting with the database for retrieving client information and creating outbound communication creation.

IV. Results

Employing the methods described above, a comprehensive data pipeline was developed (***Fig. 1)*** :

- **PostgreSQL (db):** This service sets up the PostgreSQL database.
- **Admin for PostgreSQL Management (pgadmin):** This service establishes the pgAdmin container, facilitating PostgreSQL management.
- **Database Setup (db_setup):** This service executes database setup tasks. It involves initializing the database schema, populating initial data, and preparing data for modeling.
- **Model Container (model):** This service builds the model container, which generates survival predictions, populates the database with the results, as well as hosts a Jupyter notebook containing information on EDA and modeling results.
- **API Container (app):** This service constructs the API container, specifying a command to start the FastAPI application.

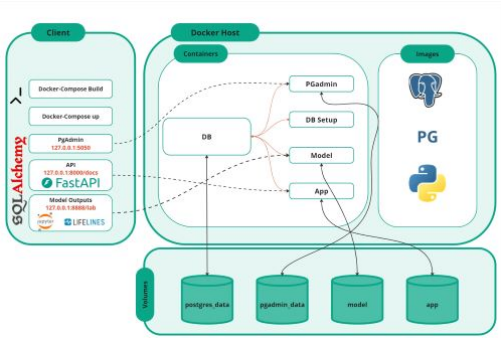

Fig. 1. Flow of Docker Containers

Let's discuss the functionalities of the flow in more detail:

A. Database Setup

The flow starts with setting up a PostgreSQL database, which can be administered through pgadmin. Once PostgreSQL is ready to accept connections, the db_setup service is initiated. In case the database is being initiated for the first time, it ensures the database schema ((***Fig. 2)***) is initiated and the original dataset (***Table 1)*** is populated. It's important to note that the data was obtained directly from the database and was already processed by the company ETL pipeline; thus, the pipeline of this project is initiated by directly inserting the data into the database. Once this process is finished, it creates and executes a procedure for generating and populating the data that will later be used for modeling customer survival. The composition of the table can be seen in (***Appendix 1 Appendix 2)*** . It is important to note that volume mapping was implemented to ensure the information in the database is not lost when the containers are stopped.

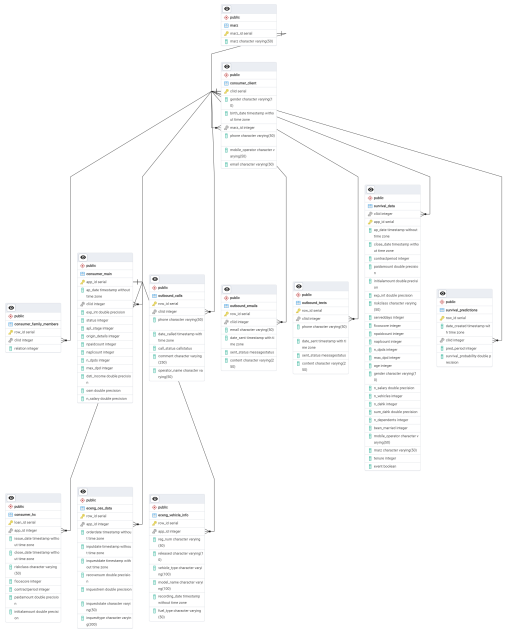

Fig. 2. Entity-Relationship Diagram

B. Modelling

Once the database setup is finished, the model container is initiated. It involves retrieving the dataset intended for modeling survival from the db, choosing an optimal distribution for the AFT model among Weibull, Log-Normal, and Log-Logistic distributions based on Akaike information criterion (AIC), fitting the model, generating customer survival predictions for a predefined interval of 30 days and adding the obtained predictions to the database.

Let's also discuss the modeling process and key findings. Initially, Exploratory Data Analysis was performed for the obtained survival data. Even though feature engineering was administered in the database setup section, some changes were employed to handle highly correlated or significantly skewed features.

Afterward, the Kaplan-Meier technique was applied, and the survival curve for the model was plotted (***Fig. 3)***. This curve illustrates the proportion of subjects surviving as a function of time, providing valuable insights into the survival experience of the population under study. In our case, the curve displays a promising trend - survival probabilities decrease with time.

Next, the main modeling technique - AFT, was applied. The lowest AIC value of 31392.700205122117 was yielded for Log-Normal distribution. Therefore, this model was chosen as the optimal one. The model was fit, and insignificant variables were removed to reduce model complexity. A summary of the obtained model can be seen in (***Fig. 4)***

1) Variable Interpretation: The coefficients of the AFT model are interpreted as follows: a unit increase in a covariate leads to a change in the average/median survival time by a factor of $\exp(\text{coef})$.

- **Contract Period:** Each additional unit of the contract

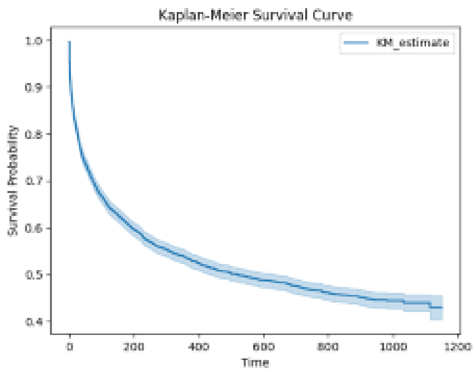

Fig. 3. Survival Curve of Kaplan-Meier

period is associated with an increase in the hazard by a factor of 1.024.

- **Interest Rate (`exp_int`):** A one-unit increase in the interest rate is associated with an increase in the hazard by a factor of 1.025.
- **FICO Score (`ficoscore`):** Each one-unit increase in the FICO score is associated with an increase in the hazard by a factor of 1.009.
- **Region (`marz_...`):** The reference group for these variables is Yerevan. Significant differences were found for the Lori and Syuniq regions compared to Yerevan, as well as for missing values. All three variables exhibit a decreasing effect on hazard compared to Yerevan. Notably, the Syuniq region showed the lowest p-value, indicating a decrease in the hazard by a factor of 0.299 compared to Yerevan.
- **Mobile Operator (`mobile_operator_...`):** The reference group for these variables is Ucom. Significant differences were observed for the Beeline and VivacellMTS operators compared to Ucom. Both variables demonstrate an increasing effect on hazard compared to Ucom, with Beeline increasing hazard by a factor of 1.47 and VivacellMTS by a factor of 1.278.
- **Number of Dependents (`n_dependents`):** Each additional dependent is associated with a decrease in the hazard by a factor of 0.735.
- **Number of DPDS (`n_dpds`):** Each additional occurrence of a missed payment is associated with an increase in the hazard by a factor of 1.392.
- **Number of Vehicles (`n_vehicles`):** Each additional vehicle is associated with an increase in the hazard by a factor of 1.316.
- **Number of Paid Loans (`npaidcount`):** Each additional paid loan in the Company is associated with a decrease in the hazard by a factor of 0.919.
- **Paid Amount (`paidamount`):** Each one-unit increase in the paid amount is associated with an increase in the hazard by a rather small factor.
- **Risk Class (`riskclass_...`):** The reference category for these variables is the Standard risk class. Being in a risk class other than Standard significantly increases hazard.

model	lifelines.LogNormalAFTFitter
duration col	'tenure'
event col	'event'
number of observations	4694
number of events observed	2366
log-likelihood	-15678.446
time fit was run	2024-05-09 18:33:36 UTC

		coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
mu_	contractperiod	0.024	1.024	0.005	0.013	0.034	1.013	1.035	0.000	4.456	<0.0005	16.868
	exp_int	0.025	1.025	0.003	0.019	0.031	1.019	1.032	0.000	8.134	<0.0005	51.099
	ficoscore	0.009	1.009	0.001	0.006	0.011	1.006	1.012	0.000	6.482	<0.0005	33.366
	marz_ԼՈՌԻ	-0.626	0.535	0.290	-1.194	-0.058	0.303	0.944	0.000	-2.161	0.031	5.024
	marz_Չնշված	-0.572	0.565	0.165	-0.895	-0.249	0.409	0.780	0.000	-3.469	0.001	10.902
	marz_ՍՅՈՒՆԻՔ	-1.208	0.299	0.380	-1.953	-0.464	0.142	0.629	0.000	-3.180	0.001	9.406
	mobile_operator_Beeline	0.386	1.471	0.170	0.052	0.719	1.053	2.053	0.000	2.266	0.023	5.413
	mobile_operator_VivacellMTS	0.246	1.278	0.141	-0.030	0.521	0.971	1.684	0.000	1.747	0.081	3.633
	n_dependents	-0.308	0.735	0.060	-0.425	-0.191	0.654	0.826	0.000	-5.165	<0.0005	21.990
	n_dpds	0.331	1.392	0.057	0.218	0.443	1.244	1.558	0.000	5.760	<0.0005	26.827
	n_vehicles	0.275	1.316	0.080	0.118	0.432	1.125	1.540	0.000	3.431	0.001	10.700
	naplcount	-0.063	0.939	0.007	-0.076	-0.050	0.927	0.951	0.000	-9.525	<0.0005	69.039
	npaidcount	-0.084	0.919	0.027	-0.137	-0.032	0.872	0.968	0.000	-3.161	0.002	9.314
	paidamount	-0.000	1.000	0.000	-0.000	-0.000	1.000	1.000	0.000	-6.131	<0.0005	30.096
	riskclass_Անհուսալի	4.732	113.534	1.122	2.534	6.930	12.602	1022.846	0.000	4.219	<0.0005	15.316
	riskclass_Կասկածելի	3.280	26.564	1.069	1.183	5.376	3.266	216.068	0.000	3.067	0.002	8.851
	riskclass_Հսկվող	2.133	8.441	0.318	1.510	2.756	4.527	15.741	0.000	6.709	<0.0005	35.574
	riskclass_Ոչ ստանդարտ	3.952	52.049	0.883	2.221	5.683	9.217	293.916	0.000	4.475	<0.0005	16.996
	Intercept	-0.065	0.937	0.820	-1.672	1.543	0.188	4.678	0.000	-0.079	0.937	0.094
sigma_	Intercept	1.158	3.184	0.016	1.127	1.189	3.086	3.284	0.000	72.966	<0.0005	inf

Concordance	0.687
AIC	31396.893
log-likelihood ratio test	1019.068 on 18 df
-log2(p) of ll-ratio test	678.434

Fig. 4. AFT Model Summary

Finally, partial effects on outcome were plotted to compare the baseline curve of the model versus what happens when a covariate is varied over values in a group and assess feature impacts (***Fig. 5***). Key findings include:

- **Contract Period (contractperiod):** The shorter the contract period of the client's previous loan, the higher the likelihood of him/her returning to take another one.
- **Interest Rate (exp_int):** The lower the interest rate of the client's previous loan, the higher the likelihood of him/her returning to take another one.
- **FICO Score (ficoscore):** The lower the FICO Score, the higher the probability of him/her taking another loan. Moreover, if the client has no previous loan history (FICO Score is 0), then he/she is significantly more likely to take another loan.
- **Number of Closed Loans (nPaidCount):** The more loans the client has paid off in the Company, the higher the likelihood that the client will take another one.
- **Number of DPDS (n_dpds):** The more times the client has missed his/her payment date, the lower the likelihood that the client will take another loan.
- **Number of Vehicles (n_vehicles):** The fewer cars the client has, the higher the likelihood that the client will take another one.
- **Paid Amount (paidamount):** The more the client has paid for his latest loan, the higher the likelihood that the client will take another one.
- **Number of Dependents (n_dependents):** The more dependent the client has, the higher the likelihood of him returning to take another loan.
- **Mobile Operator:** In comparison with the Ucom mobile operator, customers of Beeline and VivacellMTS have a lower likelihood of the event occurring.
- **Region:** In comparison with Yerevan, being from the regions of Syuniq and Lori also negatively impacts the likelihood of a client returning for another loan.
- **Risk Class:** Not having a "Standard" risk class is also associated with a lower likelihood of the client returning to take another loan.

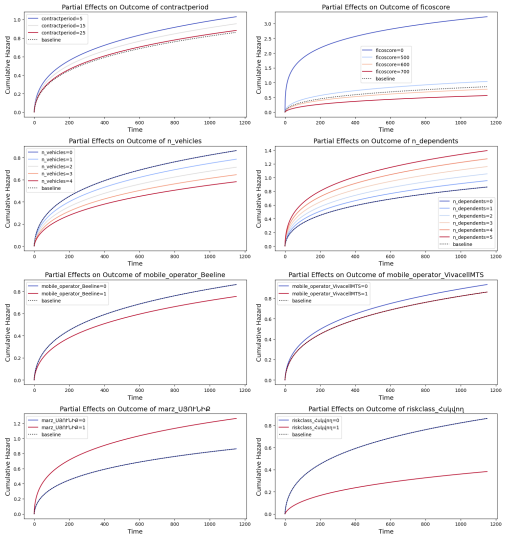

Fig. 5. Feature Effects

C. API

The final structural component of the project is the API, which is connected to the database and can either retrieve data from the database or insert new records. The API currently offers 5 actions:

- **Retrieve Client Information:** The `/get_client_info/` endpoint allows users to retrieve information about clients based on their unique client IDs (cliids).
- **Query Survival Data:** The `/get_survival_data/` endpoint enables users to query survival data based on prediction period, survival probability limits, and creation date. This functionality is particularly useful for identifying target groups for customer retention efforts.
- **Create Outbound Calls:** The `/create_call/` endpoint facilitates the creation of outbound call records in the system. Customer service operators can record the call status and comments related to their outbound call activities.
- **Create Outbound Text Messages:** The `/create_text/` endpoint allows users to generate outbound text messages for one or more clients. Users specify message details such as sent status and content.
- **Create Outbound Emails:** The `/create_email/` endpoint provides functionality for creating outbound email communications. Users provide email details, including recipient email address, sent status, and content, along with a list of client IDs (cliids) to send the emails to.

V. Conclusion

Overall, this project yielded a comprehensive data pipeline stretching from database setup, modeling, and an API. The implemented model aims to predict the survival probabilities of the clients using an AFT model, which can be utilized to optimize customer retention efforts. For example, customers with a moderate level of survival probability who are at risk of not taking another loan from the Company can be targeted by the call center operators and offered some special promotions encouraging the customer to remain with the company.

In order to capture the true effect of utilizing the infrastructure proposed in this project, it is advised to set test and control groups of clients. The control group will continue to be treated using current Company practices, whereas the test group will be targeted based on their survival predictions to assess whether the proposed model brings about statistically significant improvements to company retention levels.

References

[1] Container Deployment — pgAdmin 4 8.6 documentation. (n.d.). *Www.pgadmin.org*. Retrieved May 9, 2024, from https://www.pgadmin.org/docs/pgadmin4/latest/container_deployment.html

[2] Docker. (2020, April 9). Docker overview. *Docker Documentation*. https://docs.docker.com/get-started/overview/

[3] Docker Hub. (n.d.). *Hub.docker.com*. https://hub.docker.com/_/postgres

[4] FastAPI. (n.d.). *FastAPI*. https://fastapi.tiangolo.com/

[5] Lifelines — lifelines 0.27.0 documentation. (n.d.). *Lifelines.readthedocs.io*. https://lifelines.readthedocs.io/en/latest/

[6] Shepard, A. (2023, March). What are The Different Types of APIs and Protocols? *Kong Inc*. https://konghq.com/blog/learning-center/different-api-types-and-use-cases

VI. Appendix

Appendix 1: The Composition of Tables of Original Dataset

Table	Column	Description	Data Type
marz	marz_id	Unique identifier of each region.	Integer
marz	marz	Name of region.	Varchar
consumer_client	cliid	Unique identifier of each client.	Integer
consumer_client	gender	Gender of client.	Varchar
consumer_client	birth_date	Birth date of client.	Datetime
consumer_client	marz_id	Origin region identifier for the client.	Integer
consumer_client	phone	Phone number of the client.	Varchar
consumer_client	mobile_operator	The mobile operator of the client.	Varchar
consumer_client	email	The email of the client.	Varchar
consumer_family_members	row_id	Unique identifier of each row.	Integer
consumer_family_members	cliid	Identifier of client.	Integer
consumer_family_members	relation	Identifier of relation (3 - spouse, 5 - dependent).	Integer
consumer_main	app_id	Unique identifier of each loan application.	Integer
consumer_main	ap_date	Date of loan application.	Datetime
consumer_main	cliid	Identifier of client.	Integer
consumer_main	exp_int	Interest rate.	Float
consumer_main	status	Status of application (14 - disbursed).	Integer
consumer_main	apl_stage	Stage of application (28 - closed, 29 - active).	Integer
consumer_main	origin_details	Type of loan applied for.	Integer
consumer_main	npaidcount	Number of previously closed loans.	Integer
consumer_main	naplcount	Number of previous applications.	Integer
consumer_main	n_dpds	Number of times the client paid the past due date.	Integer
consumer_main	max_dpd	Maximum days paid past due.	Integer
consumer_main	dsti_income	Debt service-to-income (DSTI) modelled income.	Float
consumer_main	osm	Difference between total income and liabilities.	Float
consumer_main	n_salary	Officially registered salary.	Float
consumer_hc	loan_id	Unique identifier of each loan.	Integer
consumer_hc	app_id	Unique identifier of each application.	Integer
consumer_hc	issue_date	Date the loan was issued.	Datetime
consumer_hc	close_date	Date the loan was closed.	Datetime
consumer_hc	riskclass	Risk class of the client.	Varchar
consumer_hc	contractperiod	The contract period of the loan.	Integer
consumer_hc	ficoscore	FICO Score	Integer
consumer_hc	paidamount	Amount paid back.	Float
consumer_hc	initialamount	Amount disbursed.	Float
eceng_vehicle_info	row_id	Unique identifier of each row.	Integer
eceng_vehicle_info	app_id	Identifier of loan application.	Integer
eceng_vehicle_info	reg_num	Registration number of vehicle.	Varchar
eceng_vehicle_info	released	The year the vehicle was released.	Varchar
eceng_vehicle_info	model_name	The name of the model of the vehicle.	Varchar
eceng_vehicle_info	recording_date	The date the vehicle was recorded.	Datetime
eceng_vehicle_info	fuel_type	The type of fuel the car operates on.	Varchar
eceng_ces_data	row_id	Unique identifier of each row.	Integer
eceng_ces_data	app_id	Identifier of loan application.	Integer
eceng_ces_data	orderdate	The date the proceeding was ordered.	Datetime
eceng_ces_data	inputdate	The date of input of the proceeding.	Datetime
eceng_ces_data	inquestdate	The date the proceeding was inquested.	Datetime
eceng_ces_data	recoversum	The amount to be recovered.	Float
eceng_ces_data	inquestrem	The remainder amount.	Float
eceng_ces_data	inqueststate	The state of the inquest.	Varchar
eceng_ces_data	inquesttype	The type of the inquest.	Varchar

Appendix 2: The Composition of Tables introduced during the project.

survival_data	cliid	Identifier of the client.	Integer
survival_data	app_id	Identifier of the client's latest loan.	Integer
survival_data	ap_date	Date of loan application.	Datetime
survival_data	close_date	Date the loan was closed.	Datetime
survival_data	contractperiod	The contract period of the loan.	Integer
survival_data	paidamount	Amount paid back.	Float
survival_data	initialamount	Amount disbursed.	Float
survival_data	exp_int	Interest rate.	Float
survival_data	riskclass	Risk class of the client.	Varchar
survival_data	serveddays	The number of days the loan was active for.	Integer
survival_data	npaidcount	Number of previously closed loans.	Integer
survival_data	naplcount	Number of previous loan applications.	Integer
survival_data	n_dpds	Number of times the client paid the past due date.	Integer
survival_data	max_dpd	Maximum days paid past due.	Integer
survival_data	age	Age of the client.	Integer
survival_data	gender	Gender of the client.	Varchar
survival_data	n_salary	Registered salary of the client.	Float
survival_data	n_vehicles	Number of vehicles registered to the client.	Integer
survival_data	n_dahk	Number of legal proceedings of the client.	Integer
survival_data	n_dependets	Number of dependents of the client.	Integer
survival_data	been_married	Marital status (0: Never been married, 1: Has been married).	Integer
survival_data	sum_dahk	Amount (AMD) of enforcement proceedings of the client.	Float
survival_data	mobile_operator	The mobile operator.	Varchar
survival_data	tenure	Tenure, the duration between the client closing his/her latest loan (in 2021) and taking the next.	Integer
survival_data	event	Event, a boolean indicating if an event occurred (True if took another loan).	Boolean
survival_predictions	row_id	Unique identifier of each row.	Integer
survival_predictions	date_created	Date the prediction was created on.	Datetime
survival_predictions	cliid	Identifier of the client.	Integer
survival_predictions	pred_period	Prediction period.	Integer
survival_predictions	survival_probability	Survival probability.	Float
outbound_calls	row_id	Unique identifier of each row.	Integer
outbound_calls	cliid	Identifier of the client.	Integer
outbound_calls	phone	Phone number of the client.	Varchar
outbound_calls	date_called	Date the outbound call was made.	Datetime
outbound_calls	call_status	The status of the call.	Varchar
outbound_calls	comment	Additional comments.	Varchar
outbound_calls	operator_name	The name of the calling operator.	Varchar
outbound_text	row_id	Unique identifier of each row.	Integer
outbound_text	cliid	Identifier of the client.	Integer
outbound_text	phone	Phone number of the client.	Varchar
outbound_text	date_sent	Date the outbound text was sent.	Datetime
outbound_text	sent_status	The status of the message.	Varchar
outbound_text	content	The message content.	Varchar
outbound_email	row_id	Unique identifier of each row.	Integer
outbound_email	cliid	Identifier of the client.	Integer
outbound_email	email	Email of the client.	Varchar
outbound_email	date_sent	Date the outbound email was sent.	Datetime
outbound_email	sent_status	The status of the email.	Varchar
outbound_email	content	The email content.	Varchar