

# **Loan Default Prediction to Measure the Likelihood of Default by Borrowers of Consumer Loans for Online Digital Lending**

Debanjan Datta, Ilansurya Ilanchezhian, Valerian Yap,  
Singapore Management University

## **ABSTRACT**

Superlender is South African online digital lending company, which employs effective use of credit risk models to deliver profitable and high-impact loan alternatives. Assessment of credit risks are based on two main risk drivers of loan default prediction: (i) willingness to pay and (ii) ability to pay.

The research paper delves into the credit default risk model approach to determine how important variables are to predict the binary outcome of whether a customer is assigned a good flag (lower default risk likelihood) and bad flag (higher default risk likelihood). The online digital lender also works with banking partners from several key geographies globally (mainly in Nigeria and with some from banks branches located in USA, UK, UAE, China, Thailand, Russia and Australia) for the disbursement of consumer credit loans.

Our primary objective is to identify the best prediction model to categorize good or bad flag customers and potentially the credit approval process for existing-to-business customers to obtain new loans. Our analysis provides insights into how demographics data, previous loans data and existing loans data could predict whether a customer is categorized as good or bad flag based on existing data collected by the business.

This study enhanced our understanding of loan default predictors and could have a positive impact for consumer credit businesses, including banks and not limited to online digital lending players. The insights obtained could help improve the reputation of consumer credit businesses as there is generally a bad reputation that consumer loans are detrimental and should not be taken. This occurs when write-offs of bad debt for consumer credit businesses are high and unsuitable consumers approved with a consumer credit loan are declared bankrupt due to inability to repay excessive cumulative debt.

## **INTRODUCTION**

### **MOTIVATION**

Loan default prediction required by financial institutions to assess the credit worthiness of borrowers prior to the approval of a loan requested by a borrower. Accurate prediction models can mitigate risks and financial losses as it helps the financial institution avoid lending to customers with high credit risk. This safeguards both the reputation of the lending body (i.e., the financial institution) and helps customers avoid a situation of inability to service their debts.

An article published by McKinsey found that external and internal pressures are requiring banks to reevaluate the cost efficiency and sustainability of their risk-management models and processes. Some of the pressure comes, directly or indirectly, from regulators; some from investors and new competitors; and some from the banks' own customers. The impact is also being felt on the bottom line, where risk and compliance costs amounted to approximately 10% of total banking costs and return on equity remains below costs of capital due to additional capital requirements, fines and lagging costs efficiencies (McKinsey & Company, 2016).

The International Accounting Standards Board (IASB) has set out principles-based standards on how banks should recognize and provide for credit losses for financial statement reporting purposes. In July 2014, the IASB introduced an "expected credit loss" (ECL) framework for the recognition of impairment (Bank for International Settlements, 2017). To aid in such reporting, a reliable default prediction model can assist lenders to adhere to these regulations by ensuring that bad flag loans are identified and accurately provisioned for, thereby avoiding hefty fines.

Automation can allow for faster loan processing, streamlining systems, provide reliable and consistent dataflow for any stage of the loan origination process, and quicken the overall process, while delivering audit and control benefits (Moody's Analytics, 2018). Hence, automation can enhance both the experience of employees working in such financial institutions and of customers applying for new loans. Overheads could also potentially be reduced as automation helps alleviate the need for manual credit review process for all loans.

Having a faster credit review process would also provide a significant competitive advantage between SuperLender and its competitors as the primary factor that customer look at when selecting a financial institution to borrow from is the processing speed of the loan (McKinsey & Company, 2021).

### **PROBLEM STATEMENT & RESEARCH OBJECTIVE**

A loan default prediction model is a useful way to measure and identify customers with higher risks of defaulting on new loans taken with lenders. However, given that digital online lenders are relatively new business models, limited research has been conducted on online-lenders to predict loan default likelihood by individual lenders of consumer credit loans.

Therefore, in this study, we would seek to develop several predictive models and compare their performances to identify the most suitable and accurate model using SAS Viya Visual Statics and Visual Data Mining and Machine Learning.

## LITERATURE REVIEW

In recent years, the use of data mining and machine learning techniques to predict loan default has gained traction and popularity. Various studies have explored the potential of different types of data sources, including demographic, socio-economic, and psychological variables to predict academic performance and design warning systems for at-risk borrowers from different demographic and historical borrowing records.

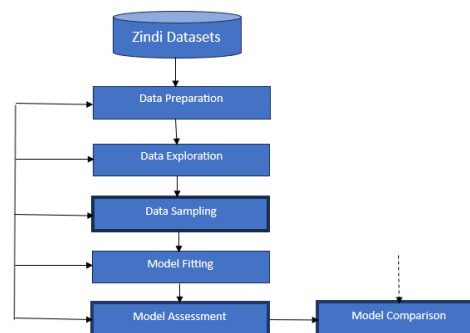
Many studies have explore the prediction of loans default through a combination of statistical (Logistic Regression) and machine learning techniques (Decision Tree, K-Nearest Neighbor, Random Forest, stochastic gradient boosting, Naïve Bayes etc.) to predict potential loans default by borrowers (Madaan, Kumar, Keshri, Jain, & Nagrath, 2020). However, in reviewing effectiveness and applications of such models in credit scoring and risk assessment a study by Breeden (2020) has concluded that it is nearly impossible to declare one best method of all.

In recent times, some researchers have also explored the use of advanced algorithms to try to improve the accuracy of credit default prediction models to increase the accuracy of loan default prediction. For example, AdaBoost model was used to achieve 100% accuracy for loan default prediction, outperforming other models including XGBoost, random forest, k-nearest neighbors and multilayer perceptrons, using real-world dataset from a prestigious international bank (Lai, 2020). In another example by Liu, Zhong, Yang, Wang & Xu (2023), the study explored the prediction of loan default in peer-to-peer lending by using the improved LightGBM and improved XGBoost algorithm to obtain the LGB-XGB-Stacking model. Results were measured by optimizing the evaluation metrics in the training phase of these two algorithms to achieve significant improvement in results especially in the recall rate of defaulted customers and the overall AUC metrics (Liu, Zhong, Yang, Wang, & Xu, 2023).

Although there are many predictive modelling studies done using machine learning techniques, there has been limited studies predicting risk of loan default at a customer level for consumer credit by an online lender specifically. Therefore, this research paper aims to fill the gap by developing an appropriate predictive model to predict the risk of loan default by each customer of SuperLender. Additionally, this research paper also aims to use SAS Viya's extreme gradient boosting CAS action ('gbtreetrain') and accompanying procedure (Gradboost) as one of the predictive models used in the study.

## METHODOLOGY

### PREDICTIVE MODELLING PROCESS



**Figure 1: Predictive Modelling Process**

## DATASET & METHODOLOGY

### DATASET

The final dataset used in this study was obtained from Loan Default Prediction Challenge (Zindi, 2023) comprising data of 4,368 customers. There were mainly three unique datasets available in the challenge, namely the train\_demographics, train\_perf and train\_prevloans.

Each of these datasets provided different information of customers of SuperLender as follows:

- **Train\_demographics (9 fields and 4,346 records/customers):** provided information of customers by birthdate, bank account type, longitude, and latitude gps, bank name, bank branches, employment status and level of education.
- **Train\_perf (10 fields and 4,368 records/customers):** provided information on current loans taken by each customer and a good or bad flag indicator to determine whether a customer has a low or high likelihood of defaulting on the loan taken.
- **Train\_prevloans (12 fields and 18,183 records / 4,359 customers):** provided information of loannumber, approveddate, creationdate, loanamount, totaldue, termdays, closeddate, firstduedate and firstrepaiddate on previous loans taken by existing-to-business customers.

## METHODOLOGY

We adopted a structured approach to derive the predictive model for the binary outcome for Loan Default Prediction, by setting up model development, validation, and evaluation flow. The dataset is split into training, validation, and test folds without sample data overlap by stratifying the dataset into 40% for training, 30% for validation and 30% for training subsets. For model development, only the training subset was used, while for model selection and comparison, only the validation and test subsets. This is to prevent the developed models from under or overfitting the data and ensuring the generalizability of the models to make accurate classifications on the data it has not been trained on.

## DATA PREPARATION & EXPLORATION

To prepare the data for predictive analytics, several steps were taken to clean and prepare each dataset prior to combining and joining all three datasets based on the common customerID field in all three datasets. The following is a summary of the steps were undertaken to the respective datasets:

- Reformatted variables to correct data types (e.g., converted datetime values from varchar to datetime format and after splitting date and time fields using SAS Studio, date was converted from varchar to date format using SAS Data Studio; extracted variable such as birthyear by extracting “year” from date in “dd-mmm-yyyy” format using SAS Data Studio).
- Excluded variables with more than 30% of missing data (e.g., “referredby” with 94.36% of missing values; “bank\_branch\_clients” with 98.82% of missing values; “level\_of\_education\_clients” with 86.48% of missing values).
- Excluded irrelevant variables such as metadata and data not used (e.g., “loannumber” as in our data preparation process, it was discovered that “loannumber” did not accurately capture the frequency of previous loans by each customer as some previous loans were not captured in the dataset (*refer to Appendix C – Data Preparation Log*)).
- Excluded variables with highly similar information or interim variables used to derive certain final variables (e.g., was not used)

## FINAL DATASET (DATA SAMPLING)

The final dataset contains 4,368 rows, with the training, validation and test subsets comprising 40% (1,747 observations), 30% (1,311 observations) and 30% (1,310 observations) of data respectively. The variables selected to be used for predictive modelling are detailed below:

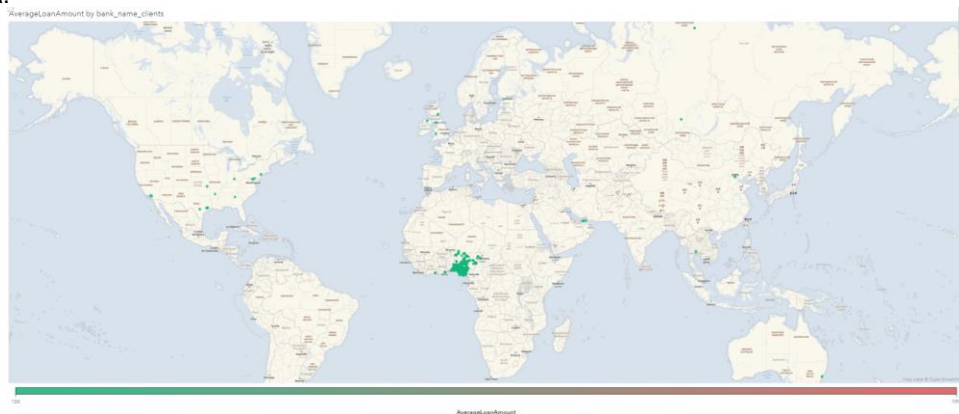
S/N	Variables	Data Type	Description	Data Source
1	CustomerID	Nominal	ID of every customer of SuperLender	Original
2	Recency	Continuous	Denotes how recent the existing loan was taken (smaller no. indicates most recent)	Derived
3	PREVRecency	Continuous	Denotes how recent the previous loans was taken (smaller no. indicates most recent)	Derived
4	PREVFrequency	Continuous	Count of the no. of loans taken in the past, excluding existing loan (if any)	Derived

5	AverageLoanAmount	Continuous	Existing loan amount in USD	Derived
6	AveragePREVLoanAmount	Continuous	Average of past loan amounts taken in USD	Derived
7	AverageTotalDue	Continuous	Existing total due amount in USD	Derived
8	AveragePREVTotalDue	Continuous	Average of past total due amounts in USD	Derived
9	AveragePREVInterest	Continuous	Average of interest on past loans taken calculated based on an annualized basis using $(\text{term days} / 365) * ((\text{AverageTotalDue} - \text{AverageLoanAmount}) / \text{AverageLoanAmount})$	Derived
10	PREVAvgDaystoRepayLoan	Continuous	Average of days to repay loan calculated as a difference between ClosedDate and CreationDate	Derived
11	PREVAvgLatePayment	Continuous	Average days where no. of days to repay loan exceeded termdays	Derived
12	BirthYear	Continuous	Birth year of each customer derived from Birthdate of each customer	Derived
13	PrevAvgLateFirstPayDate	Continuous	Average days where FirstRepaidDate exceeded FirstDueDate. Values $\geq 0$ indicates that the loan was paid prior to due date, while values $< 0$ indicate late first payments.	Derived
14	Bank_account_type	Nominal	Type of primary bank account	Original
15	Longitude_gps	Nominal	Longitude of the bank branch location (based on World Geodetic System (WGS84)	Original
16	Latitude_gps	Nominal	Latitude of the bank branch location (based on World Geodetic System (WGS84)	Original
17	Bank_name_clients (3,264 rows)	Nominal	Name of the bank	Original
18	Employment_status_clients (2,771 rows only)	Nominal	Type of employment that customer has	Original
19	Good_bad_flag	Nominal	Good = settled loan on time; bad = did not settled loan on time) - this is the target variable that we need to predict	Original

## ANALYSIS AND RESULTS

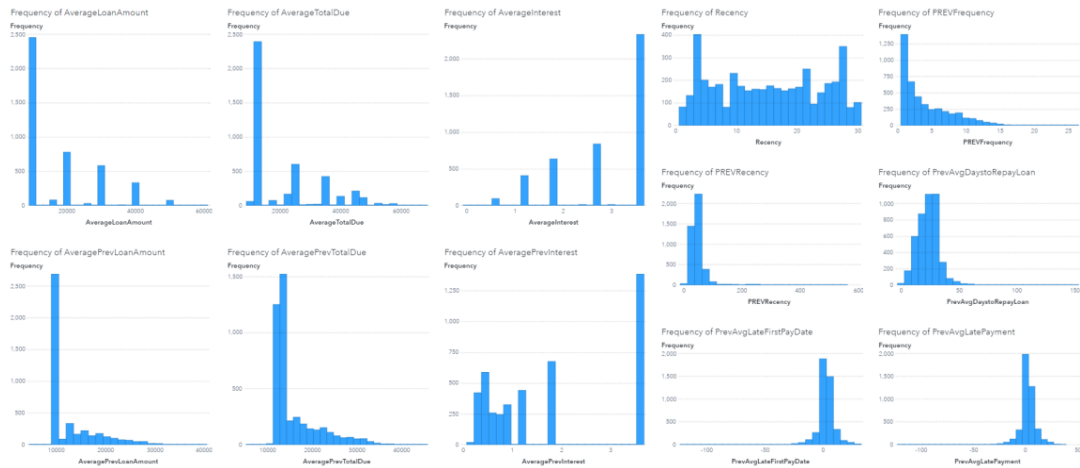
### UNIVARIATE & DESCRIPTIVE ANALYSIS

Using longitude and latitude GPS data, we noticed that most customers in our dataset has taken loans from banks based in Africa, particularly in Nigeria, some from banks located in USA and a few in UK, UAE, China, Thailand, Russia and Australia.



**Figure 2: Location of bank branches**

Univariate analysis was conducted to identify scenarios that may lead to complete separation scenario. We identified that some of these variables are skewed and may potentially require further transformation prior to statistical modeling.



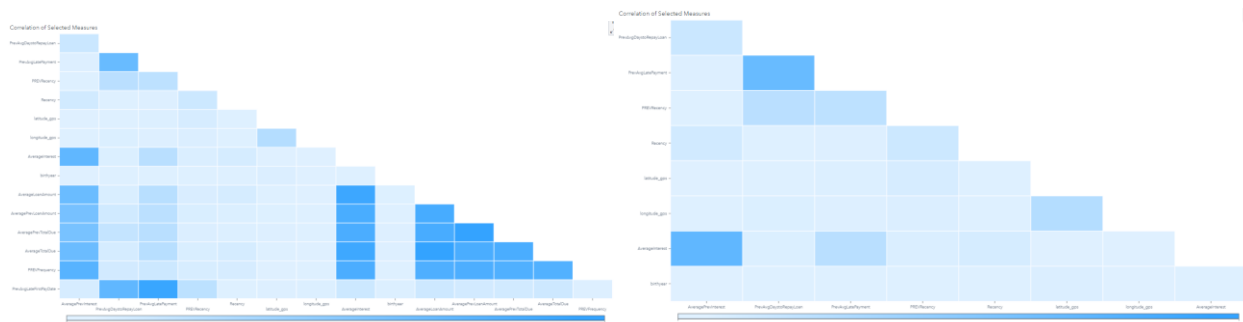
**Figure 3: Histograms of individual key variables**

## BIVARIATE ANALYSIS



**Figure 4: Identified variable to drop highlighted**

We further conduct bivariate analysis for the variables in the final data and develop a correlation matrix to observe correlation between variables to detect for multicollinearity.



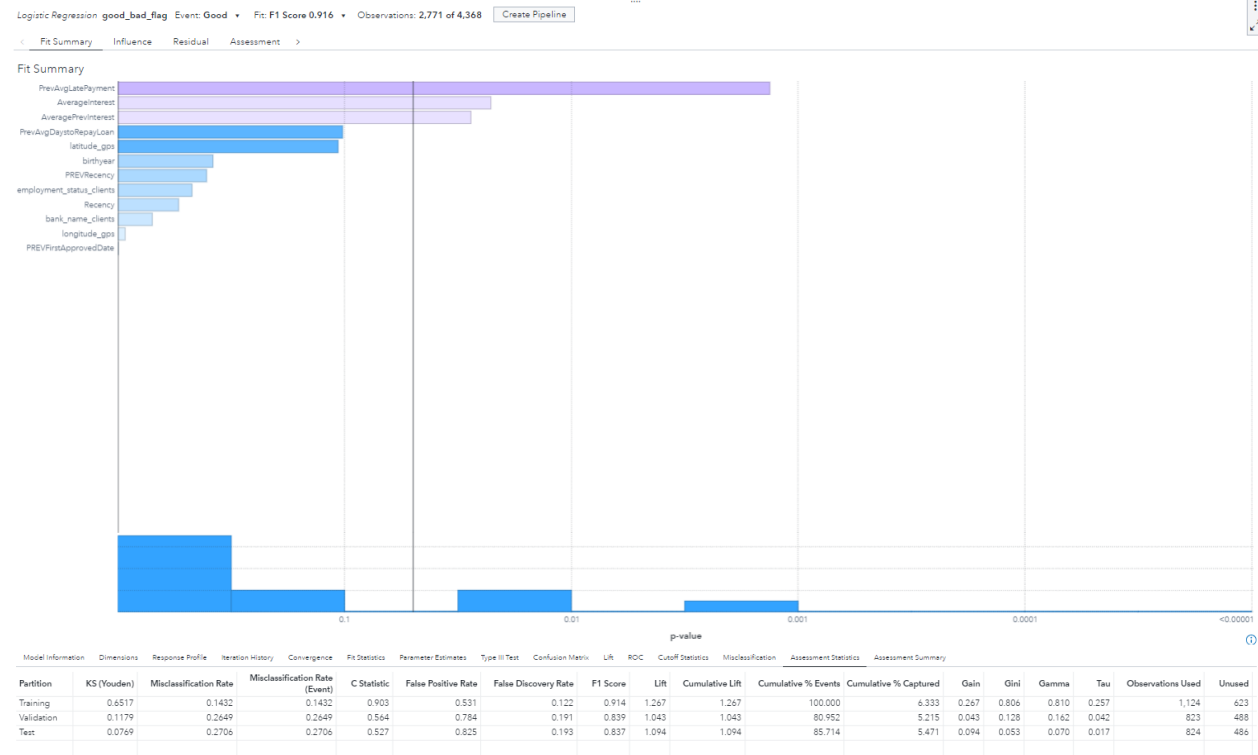
**Figure 5: Correlation Matrix to check for high correlation (>0.8). Matrix (left) refers to using all selected variables, while matrix (right) shows variables post removal of highly correlated variables.**

We removed variables “AverageLoanAmount”, “AveragePrevLoanAmount”, “AverageTotalDue”, “AveragePrevTotalDue”, “PrevAvgLateFirstPay” and “PrevFrequency” (as highlighted in Figure 4) since these

variables are highly correlated ( $>0.8$ ) with other variables, resulting in our selected variables in the correlation matrix with reduced variables (right of Figure 5) for further analysis.

## STATISTICAL MODELING TECHNIQUES

For statistical modelling, we focused on using the logistic regression model, which is a type of statistical modelling technique suitable for predicting binary outcomes (Fritz & Berger, 2015).



**Figure 6: Logistic Regression Fit Summary**

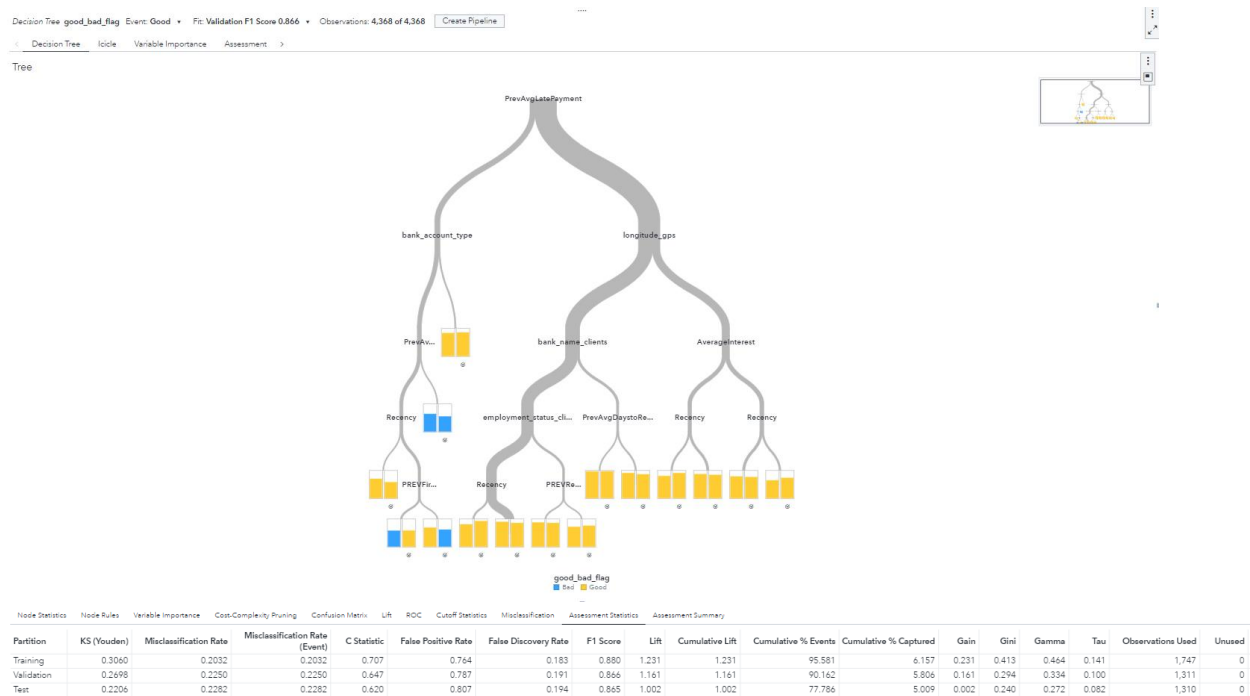
From our logistic regression analysis, a significance level of 0.05 was kept and (i) PrevAvgLatePayment, (ii) AverageInterest and (iii) AveragePrevInterest came out as predictors that were statistically significant at that level of significance with a high training F1 score of 0.901, which indicates that customers who had previous late payments and higher costs of interests for existing & previous loans had a higher likelihood of default.

## MACHINE LEARNING TECHNIQUES

In terms of machine learning predictive models, we used three machine learning techniques that does recursive partitioning – Decision Tree, Random Forest, and Gradient Boosting.

**Decision Tree** is a machine learning technique that can handle missing values and allows both response and predictors to be continuous or categorical. In our study, since our response variable is a binary outcome, we use decision trees for partitioning based on a user-defined prediction cut-off of 0.5 using the partitioning function of SAS Viya. This would allow us to explore relationships between predictor variables and the response variable.

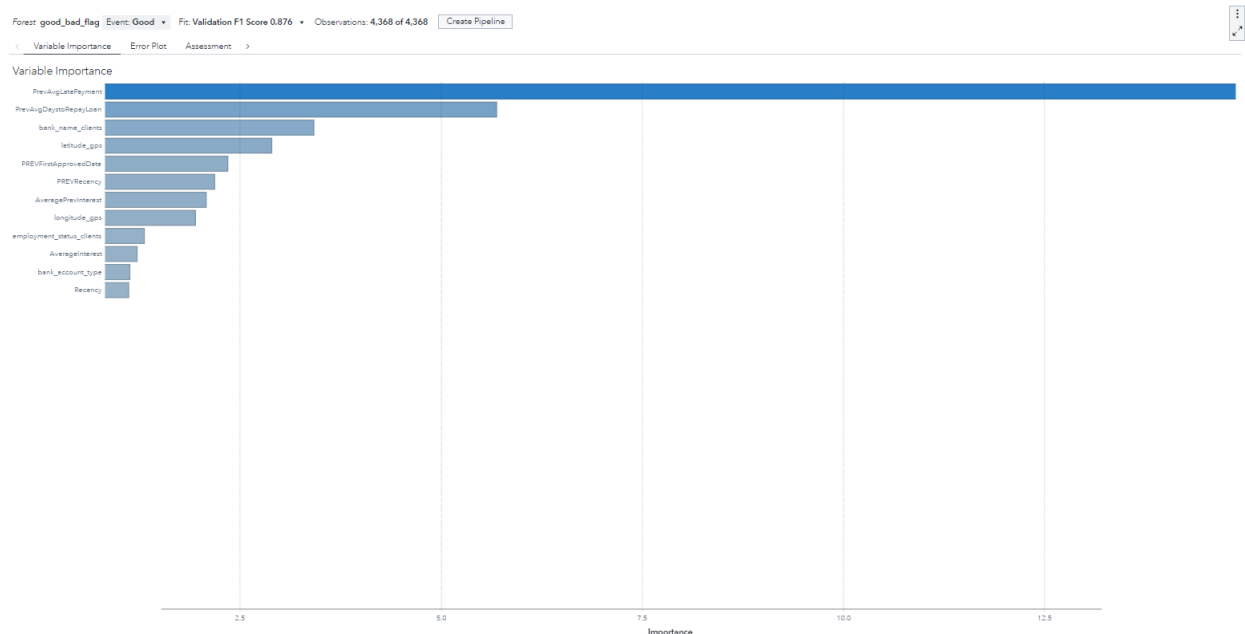
Using our final variables, our Decision Model had a training F1 score of **0.878** and a validation F1 score of **0.872**. There is a marked reduction in bias and variance compared to the logistic regression. Even with a training score that is lower than logistic regression, due to its lower overfitting tendency, it is a preferred model over logistic regression.



**Figure 7: Decision Tree Fit Summary**

**Random Forest** is machine learning technique based on decision tree and bagging (or bootstrap aggregation), an ensemble technique for improving robustness of forecasts, which generally outperforms a single Decision Tree as it tends to reduce overfitting, decrease variance, and has increased robustness especially for datasets with higher complexity, such as the datasets that we have. Therefore, in our study we seek to compare Random Forest with other models to determine the best fit model.

The Random Forest did observe minor improvement in the training and validation score compared to the Decision Tree. With a training F1 score of **0.879** and a validation score of **0.875**. Based on our results, it is the best model so far as it has low variance in training & validation F1 scores and a higher validation F1 score compared to the Decision Tree.

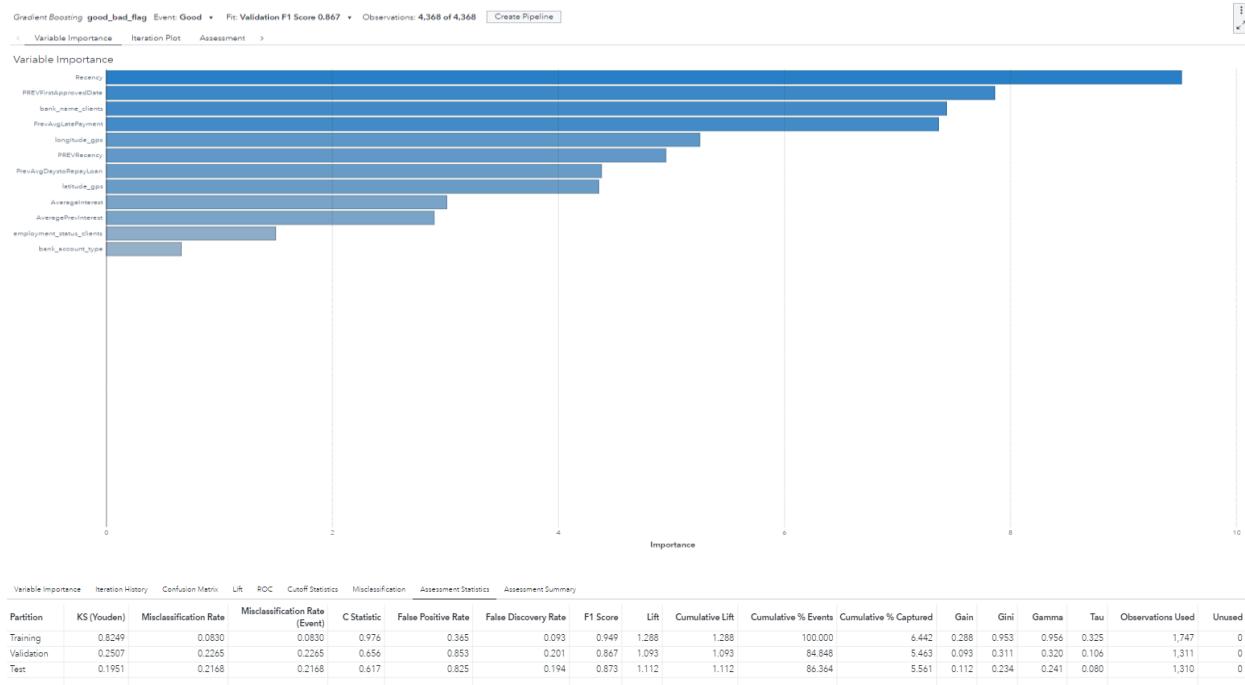


Variable Importance		Error Metric	Confusion Matrix	Lift	ROC	CutOff Statistics	Misclassification	Assessment Statistics		Assessment Summary							
Partition	KS (Youden)	Misclassification Rate	Misclassification Rate (Event)	C Statistic	False Positive Rate	False Discovery Rate	F1 Score	Lift	Cumulative Lift	Cumulative % Events	Cumulative % Captured	Gain	Gini	Gamma	Tau	Observations Used	Unused
Training	0.2401	0.1986	0.1986	0.637	0.882	0.199	0.886	1.078	1.078	83.656	5.389	0.078	0.274	0.495	0.094	1,747	0
Validation	0.1678	0.2166	0.2166	0.578	0.930	0.209	0.876	1.044	1.044	81.048	5.219	0.044	0.155	0.324	0.053	1,311	0
Test	0.1722	0.2031	0.2031	0.576	0.909	0.203	0.884	1.042	1.042	80.917	5.210	0.042	0.152	0.291	0.052	1,310	0

**Figure 8: Random Forest Feature Importance**

**Gradient Boosting** is a machine learning algorithm that builds decision trees sequentially. Each tree in the sequence aims to rectify the mistakes made by trees before. The boosting process halts when either a predefined stopping condition is satisfied or when there's no notable enhancement in model performance.

For our dataset, Gradient Boosting has a higher training F1 score with a value of **0.894** while the validation score remains the same at **0.875**. There is an increase in variance although it is not very large, which makes this model



**Figure 9: Gradient Boosting Feature Importance**

## DISCUSSION OF RESULTS & INTERPRETATIONS

### MODEL EVALUATION

The train, validation and test scores have been computed using two classification metrics – F1 score, Misclassification Rate. In determining the best statistical or machine learning prediction model for our loans default prediction model, we have selected the F1 score as the key metric to determine if the model is the best prediction model for our problem. Even though both F1 score and misclassification rate (MR) measures the rate of wrong predictions (false positives and false negatives) made by each selected model, we chose to use the F1 score as false positive and false negative rates are very different (approximately 80% false positive and 20% false negative rates) for the various types of prediction models that we have selected.

For SuperLender, granting a loan to a customer that has higher likelihood to default (false positive) and rejecting a loan request from a potential customer with low likelihood to default (false negative) would be detrimental to future business as well given that a customer may find out from another competitor lender that they are actually eligible for a loan of the same amount, but not eligible to obtain a loan from SuperLender due to a misclassification by the loan default prediction model. For the F1 score, the higher the score the better the prediction model, whereas for MR, the lower the MR the better the prediction model. Equations are given below:

$$(1) \quad Precision = \frac{TP}{TP+FP}$$



$$(2) \quad Recall = \frac{TP}{TP+FN}$$

$$(3) \quad MR = \frac{FP+FN}{TN+TP+FP+FN}$$

$$(4) \quad F1 = \frac{2*(Recall*Precision)}{(Recall+Precision)}$$

The training and validation F1 and MR scores for all four models have been displayed below followed by a detailed metric table consisting of other metrics as well for training, validation, and test data.

	Training F1 Score	Validation F1 Score	Training MR	Validation MR
<b>Logistic Regression</b>	0.901	0.850	0.165	0.249
<b>Decision Tree</b>	0.878	0.872	0.208	0.219
<b>Random Forest</b>	0.879	0.875	0.210	0.217
<b>Gradient Boosting</b>	0.894	0.875	0.183	0.217

Selected	Model	Visualization Type	Observations	Percentile	Prediction cutoff	C Statistic	Cumulative % Captured	Cumulative % Events	Cumulative Lift	F1 Score	Discovery Rate	False Positive Rate	Gain	Gamma	Gini	KS (Youden)	Lift	Misclassification Rate (Event)
No	Logistic regression - good_bad_flag 1	Logistic Regression	1,329	5	0.5	0.871	6.424	100.000	1.2848	0.901	0.146	0.594	0.285	0.748	0.743	0.579	1.2848	0.165
No	Logistic regression - good_bad_flag 1 (validation)	Logistic Regression	975	5	0.5	0.598	5.650	87.755	1.1301	0.850	0.197	0.785	0.130	0.226	0.196	0.176	1.1301	0.249
No	Logistic regression - good_bad_flag 1 (test)	Logistic Regression	960	5	0.5	0.569	5.607	87.500	1.1215	0.855	0.203	0.834	0.121	0.163	0.138	0.140	1.1215	0.244
Yes	Forest - good_bad_flag 1	Forest	1,747	5	0.5	0.625	5.364	83.260	1.0728	0.879	0.201	0.882	0.073	0.505	0.250	0.233	1.0728	0.210
Yes	Forest - good_bad_flag 1 (validation)	Forest	1,311	5	0.5	0.612	5.322	82.651	1.0644	0.875	0.205	0.899	0.064	0.463	0.223	0.211	1.0644	0.217
Yes	Forest - good_bad_flag 1 (test)	Forest	1,310	5	0.5	0.591	5.258	81.652	1.0515	0.882	0.201	0.891	0.052	0.412	0.182	0.174	1.0515	0.206
No	Decision tree - good_bad_flag 1	Decision Tree	1,747	5	0.5	0.724	6.254	97.072	1.2507	0.878	0.189	0.803	0.251	0.493	0.447	0.334	1.2507	0.208
No	Decision tree - good_bad_flag 1 (validation)	Decision Tree	1,311	5	0.5	0.637	5.610	87.121	1.1220	0.872	0.199	0.850	0.122	0.307	0.274	0.247	1.1220	0.219
No	Decision tree - good_bad_flag 1 (test)	Decision Tree	1,310	5	0.5	0.625	5.505	85.488	1.1009	0.874	0.197	0.846	0.101	0.280	0.250	0.222	1.1009	0.216
No	Gradient boosting - good_bad_flag 1	Gradient Boosting	1,747	5	0.5	0.823	6.442	100.000	1.2884	0.894	0.182	0.785	0.288	0.665	0.645	0.481	1.2884	0.183
No	Gradient boosting - good_bad_flag 1 (validation)	Gradient Boosting	1,311	5	0.5	0.631	5.459	87.879	1.1317	0.875	0.204	0.888	0.132	0.272	0.262	0.205	1.1317	0.217
No	Gradient boosting - good_bad_flag 1 (test)	Gradient Boosting	1,310	5	0.5	0.630	5.756	89.394	1.1512	0.881	0.197	0.863	0.151	0.271	0.260	0.225	1.1512	0.207

Figure 10: Classification Metric Table

## HYPERPARAMETER TUNING

Post our initial results, hyperparameters were tuned for optimized outcomes:

	Default Hyperparameters	Optimized Hyperparameters
<b>Random Forest</b>	<ul style="list-style-type: none"> <li>50 trees</li> <li>bootstrap of 0.6</li> </ul>	<ul style="list-style-type: none"> <li>1,000 trees (max)</li> <li>bootstrap of 0.8</li> </ul>
<b>Gradient Boosting</b>	<ul style="list-style-type: none"> <li>50 trees</li> <li>Learning rate of 0.1</li> </ul>	<ul style="list-style-type: none"> <li>10,000 trees (max)</li> <li>Learning rate of 0.75</li> </ul>

## MODEL COMPARISON

	Training F1 Score	Validation F1 Score	Test F1 Score
<b>Optimized Random Forest</b>	0.900	0.881	0.883
<b>Optimized Gradient Boosting</b>	0.882	0.882	0.883
<b>Random Forest</b>	0.879	0.875	0.882
<b>Gradient Boosting</b>	0.894	0.875	0.881
<b>Decision Tree</b>	0.878	0.872	0.874
<b>Logistic Regression</b>	0.901	0.850	0.855

The variation of F1 score values across training, validation and test folds were less than 5%, indicating that the risk of overfitting is minimal.

Both Optimized Random Forest & Gradient Boosting models have similar test F1 scores (0.883), indicating comparable performance when applied to new and unseen data. However, training & validation F1 scores resulted in different conclusions. Nonetheless, our final comparison results shows that in the event of further tuning of hyperparameters, Optimized Gradient Boosting can make equally good prediction model for loan default prediction with lower variance across datasets.

## CONCLUSION

Overall, our models had high F1-scores of more than 0.85 across the 4 models being tested, which indicates high levels of predictability for likelihood for default. Despite our initial results of Random Forest being the best model, tuning of

hyperparameters result in a different conclusion as Optimized Gradient Boosting is a better model compared to Random Forest in our scenario where a low variance in F1 score is preferred.

From Random Forest, Decision Tree & Logistic Regression, PrevAvgLatePayment is a key variable that had a strong importance in loan default prediction. Other factors predicted by Gradient Boosting included (i) Average Interest of Existing & Previous Loans (ii) Recency of Transactions (iii) First Approved Date of Previous Loans (iv) Related Bank, which are key takeaways from our predictive model.

## **FUTURE WORK**

One area we would like to recommend for improvement is to further improve the tuning of hyperparameters. Using other hyperparameter settings may result in different results, which may be contrary to our conclusion. Parameters such as the number of trees, the maximum depth of each tree and the minimum number of samples required to split internal nodes could be tuned.

SuperLender could also improve the data collection process to obtain better quality datasets for analysis. Defining clear data collection objectives and requiring data fields in the demographic datasets to be compulsory inputs could enable more variables can be selected for predictive analysis.

SuperLender could also consider collecting more data over a longer period to obtain more information on historical loans given that there were nine loan transactions that were new-to-business customers or customers with no existing loan records with SuperLender.

**(Word count:** 3,288 words without titles and headers)

## REFERENCES

- Bank for International Settlements. (13 December, 2017). *IFRS 9 and expected loss provisioning - Executive Summary*. Retrieved from <https://www.bis.org/fsi/fsisummaries/ifrs9.htm>
- Fritz, M., & Berger, P. (2015). Improving the User Experience through Practical Data Analytics. In M. Fritz, & P. D. Berger, *Chapter 1 - Introduction to a variety of useful statistical ideas and techniques* (pp. 1-45). Elsevier Inc. doi:<https://doi.org/10.1016/C2013-0-18588-1>
- Lai, L. (2020). Loan Default Prediction with Machine Learning Techniques. *2020 International Conference on Computer Communication and Network Security (CCNS)* (pp. 5-9). Xi'an, China: IEEE. doi:[10.1109/INCET49848.2020.9154100](https://doi.org/10.1109/INCET49848.2020.9154100)
- Liu, Z., Zhong, Z., Yang, H., Wang, G., & Xu, Z. (2023). An innovative model fusion algorithm to improve the recall rate of peer-to-peer lending default customers. *Intelligent Systems with Applications*, 20. Retrieved from <https://www.sciencedirect.com/science/article/pii/S2667305323000972#se0020>
- Madaan, M., Kumar, A., Keshri, C., Jain, R., & Nagrath, P. (2020). Loan default prediction using decision trees and random forest: a comparative study. *IOP Conference Series: Material Science & Engineering*. 1022, pp. 1-13. Rajpura, India: IOP. doi:[10.1088/1757-899X/1022/1/012042](https://doi.org/10.1088/1757-899X/1022/1/012042)
- Mckinsey & Company. (21 July, 2016). The value in digitally transforming credit risk management. Madrid, Berlin, New York, Spain, Germany, USA. Retrieved from <https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-value-in-digitally-transforming-credit-risk-management>
- Mckinsey & Company. (14 June, 2021). The coming opportunity in consumer lending. (R. Bucci, Ed.) Delhi, Beline and USA, India, Germany and New York. Retrieved from <https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/the-coming-opportunity-in-consumer-lending>
- Moody's Analytics. (November, 2018). *Moody's Analytics Enterprise Risk*. Retrieved from <https://www.moodyanalytics.com/articles/2018/maximize-efficiency-how-automation-can-improve-your-loan-origination-process>
- Zindi. (2023). *Loan Default Prediction Challenge*. Retrieved from <https://zindi.africa/competitions/data-science-nigeria-challenge-1-loan-default-prediction>

## APPENDIX A: METADATA

### Demographics Data

This are demographics data for existing customers of SuperLender.

No.	Variables	Description
1	customerid	Primary key used to merge to other data
2	birthdate	date of birth of the customer
3	bank_account_type	type of primary bank account
4	longitude_gps	Longitude of the bank branch location (based on World Geodetic System (WGS84)
5	latitude_gps	Latitude of the bank branch location (based on World Geodetic System (WGS84)
6	bank_name_clients	name of the bank
7	bank_branch_clients	location of the branch - not compulsory - so missing in a lot of the cases
8	employment_status_clients	type of employment that customer has
9	level_of_education_clients	highest level of education

### Performance (or Existing Loans) Data

This is the repeat loan that the customer has taken for which we need to predict the performance of. Basically, we need to predict whether this loan would default given all previous loans and demographics of a customer.

No.	Variables	Description
1	customerid	Primary key used to merge to other data
2	systemloanid	The id associated with the particular loan. The same customerid can have multiple systemloanid's for each loan he/she has taken out
3	loannumber	The number of the loan that you have to predict
4	approveddate	Date that loan was approved
5	creationdate	Date that loan application was created
6	loanamount	Loan value taken
7	totaldue	Total repayment required to settle the loan - this is the capital loan value disbursed + interest and fees
8	termdays	Term of loan
9	referredby	customerid of the customer that referred this person - is missing, then not referred
10	good_bad_flag	good = settled loan on time; bad = did not settled loan on time) - this is the target variable that we need to predict

### Previous Loans Data

This dataset contains all previous loans that the customer had prior to the loan above that we want to predict the performance of. Each loan will have a different systemloanid, but the same customerid for each customer.

No.	Variables	Description
1	customerid	Primary key used to merge to other data
2	systemloanid	The id associated with the particular loan. The same customerid can have multiple systemloanid's for each loan he/she has taken out
3	loannumber	The number of the loan that you have to predict
4	approveddate	Date that loan was approved

5	creationdate	Date that loan application was created
6	loanamount	Loan value taken
7	totaldue	Total repayment required to settle the loan - this is the capital loan value disbursed + interest and fees
8	termdays	Term of loan
9	closeddate	Date that the loan was settled
10	referredby	customerId of the customer that referred this person - is missing, then not referred
11	firstduedate	Date of first payment due in cases where the term is longer than 30 days. So in the case where the term is 60+ days - then there are multiple monthly payments due - and this dates reflects the date of the first payment
12	firstrepaiddate	Actual date that he/she paid the first payment as defined above


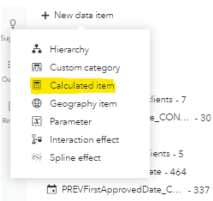
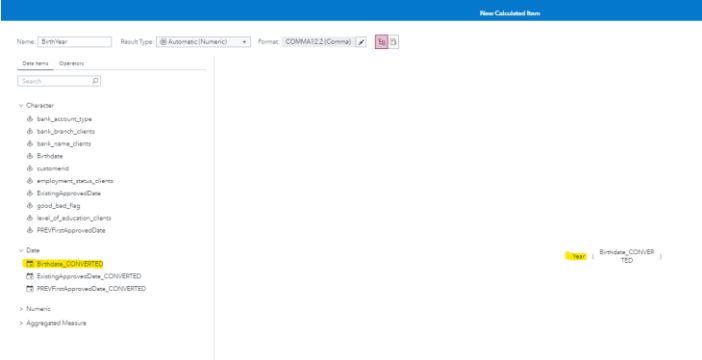
## APPENDIX B: DERIVED FINAL VARIABLES

The finalised dataset contains 4,359 rows, with the training, validation and test subsets comprising 40%, 30% and 30% of data respectively. The variables selected to be used for predictive modelling exercise are detailed below.

S/N	Variables	Formula and Details
1	Recency	<p>By Sorting Max (Date) in Descending Order = 21030</p> <p>In SAS Data Studio we would code as 21031 – LastDate to obtain latest recency of existing loans dataset.</p> <pre> 41  data RM1; 42  set RM; 43  Recency = 21031 - ExistingApprovedDate; 44  format ExistingApprovedDate date11.; 45  run; </pre> <p>Records with value 1 indicates that the transaction was just a day ago.</p>
2	PREVRecency	<p>By Sorting Max (Date) in Descending Order = 21028</p> <p>In SAS Data Studio we would code as 21029 – PREVLastDate to obtain latest recency from previous loans dataset.</p> <pre> 75  data PREVRM1; 76  set PREVRM; 77  PREVRecency = 21029 - PREVLastDate; 78  format PREVFirstApprovedDate date11.; 79  format PREVLastDate date11.; 80  run; </pre> <p>Records with value 1 indicates that the transaction was just a day ago.</p>
3	PREVFrequency	<p>Frequency is calculated as count of time for all loan transactions by each customer from previous loans dataset using code below to obtain frequency of all prior loan transactions made by each customer.</p> <pre> 75  proc sql; 76  create table PREVFreq as 77  select CustomerID, max(loannumber) as MaxLoanNumber, 78  DaystoRepayLoan, LatePayment, LateFirstPaymentDate, 79  count(distinct Time) as PREVFrequency 80  from work.PREVloans2 81  group by CustomerID; 82  quit; </pre>

4	AverageLoanAmount	<p>AverageLoanAmount is calculated by dividing provided variables LoanAmount by Frequency from performance dataset</p> <pre> 47 proc sql; 48 create table RFM as 49 select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, Recency, 50 Frequency, ExistingApprovedDate, 51 LoanAmount/Frequency as AverageLoanAmount, 52 TotalDue/Frequency as AverageTotalDue, 53 AnnualInterest/Frequency as AverageInterest 54 from RM1 as a left join Freq as b 55 on a.CustomerID = b.CustomerID; 56 quit; </pre>
5	AveragePREVLoanAmount	<p>AveragePREVLoanAmount is calculated by dividing provided variables LoanAmount by PREVFrequency from previous loan dataset</p> <pre> 111 proc sql; 112 create table PREVRFM as 113 select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, PREVRecency, 114 PREVFrequency, PREVFirstApprovedDate, 115 LoanAmount/PREVFrequency as AveragePrevLoanAmount, 116 TotalDue/PREVFrequency as AveragePrevTotalDue, 117 AnnualInterest/PREVFrequency as AveragePrevInterest, 118 PrevAvgDaystoRepayLoan, 119 PrevAvgLatePayment, 120 PrevAvgLateFirstPayDate 121 from PREVRM1 as a left join PREVFreq1 as b 122 on a.CustomerID = b.CustomerID; 123 quit; </pre>
6	AverageTotalDue	<p>AverageTotalDue is calculated by dividing provided variables TotalDue by Frequency from performance dataset</p> <pre> 47 proc sql; 48 create table RFM as 49 select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, Recency, 50 Frequency, ExistingApprovedDate, 51 LoanAmount/Frequency as AverageLoanAmount, 52 TotalDue/Frequency as AverageTotalDue, 53 AnnualInterest/Frequency as AverageInterest 54 from RM1 as a left join Freq as b 55 on a.CustomerID = b.CustomerID; 56 quit; </pre>
7	AveragePREVTotalDue	<p>AveragePREVTotalDue is calculated by dividing provided variables TotalDue by PREVFrequency from previous loan dataset</p> <pre> 111 proc sql; 112 create table PREVRFM as 113 select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, PREVRecency, 114 PREVFrequency, PREVFirstApprovedDate, 115 LoanAmount/PREVFrequency as AveragePrevLoanAmount, 116 TotalDue/PREVFrequency as AveragePrevTotalDue, 117 AnnualInterest/PREVFrequency as AveragePrevInterest, 118 PrevAvgDaystoRepayLoan, 119 PrevAvgLatePayment, 120 PrevAvgLateFirstPayDate 121 from PREVRM1 as a left join PREVFreq1 as b 122 on a.CustomerID = b.CustomerID; 123 quit; </pre>
8	AveragePREVInterest	<p>A new variable "AnnualInterest" calculated using the formula:  <math>(TotalDue - LoanAmount) / LoanAmount / (termdays/365)</math> using code below:</p> <pre> 3 proc sql; 4 create table PREVloans as 5 select distinct CustomerID, systemloanid, loannumber, 6 approveddate_CONVERTED, creationdate_CONVERTED, 7 loanamount, totaldue, termdays, closeddate_CONVERTED, 8 referredby, firstduedate_CONVERTED, firstrepaiddate_CONVERTED, 9 (totaldue - loanamount) / loanamount / (termdays/365) as annualinterest 10 from CASUSER.Trainprevloans_1 11 group by CustomerID; 12 run; </pre> <p>"AnnualInterest" is then summed up for each unique customer based on PrevLastDate of their last transaction using code below:</p>

		<pre> 94 proc sql; 95 create table PREVRM as 96 select CustomerID, sum(LoanAmount) as LoanAmount, sum(totaldue) as Totaldue, 97 max('AnnualInterest'n) as AnnualInterest, 98 max('ApprovedDate'n) as PREVLastDate, 99 min('ApprovedDate'n) as PREVFirstApprovedDate 100 from work.PREVLANS1 101 group by CustomerID; 102 quit; </pre> <p>AveragePREVInterest is calculated by dividing derived variables AnnualInterest by PREVFrequency</p> <pre> 111 proc sql; 112 create table PREVRFM as 113 select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, PREVRecency, 114 PREVFrequency, PREVFirstApprovedDate, 115 LoanAmount/PREVFrequency as AveragePrevLoanAmount, 116 TotalDue/PREVFrequency as AveragePrevTotalDue, 117 AnnualInterest/PREVFrequency as AveragePrevInterest, 118 PrevAvgDaystoRepayLoan, 119 PrevAvgLatePayment, 120 PrevAvgLateFirstPayDate 121 from PREVRM1 as a left join PREVFreq1 as b 122 on a.CustomerID = b.CustomerID; 123 quit; </pre>
9	PREVAvgDaystoRepayLoan	<p>DaystoRepayLoan per loan transaction was calculated by deducting provided variables ClosedDate from CreationDate from previous loans dataset</p> <pre> 14 data PREVLANS1; 15 set work.PREVLANS1; 16 datemonthyear = datepart(approveddate_CONVERTED); 17 time = timepart(approveddate_CONVERTED); 18 format datemonthyear date11. time time.; 19 month = month(datemonthyear); 20 day = day(datemonthyear); 21 year = year(datemonthyear); 22 ApprovedDate = mdy(month, day, year); 23 format ApprovedDate date11.; 24 drop datemonthyear month day year; 25 26 datemonthyear = datepart(creationdate_CONVERTED); 27 time = timepart(creationdate_CONVERTED); 28 format datemonthyear date11. time time.; 29 month = month(datemonthyear); 30 day = day(datemonthyear); 31 year = year(datemonthyear); 32 CreationDate = mdy(month, day, year); 33 format CreationDate date11.; 34 drop datemonthyear month day year; 35 36 datemonthyear = datepart(closeddate_CONVERTED); 37 time = timepart(closeddate_CONVERTED); 38 format datemonthyear date11. time time.; 39 month = month(datemonthyear); 40 day = day(datemonthyear); 41 year = year(datemonthyear); 42 ClosedDate = mdy(month, day, year); 43 format ClosedDate date11.; 44 drop datemonthyear month day year; 45 46 DaystoRepayLoan = ClosedDate - CreationDate; 47 LatePayment = Termdays - DaystoRepayLoan; 48 49 run; </pre> <p>PREVAvgDaystoRepayLoan was then derived using derived variables DaystoRepayLoan with PREVFrequency using code below:</p>

		<pre> 84 proc sql; 85 create table PREVFreq1 as 86 select distinct CustomerID, PREVFrequency, 87 sum(daystorepayloan) / PREVfrequency as PrevAvgDaystoRepayLoan, 88 sum(Latepayment) / PREVfrequency as PrevAvgLatePayment, 89 sum(LateFirstPaymentDate / PREVfrequency) as PrevAvgLateFirstPayDate 90 from work.PREVFreq 91 group by CustomerID; 92 quit; -- </pre>
10	PREVAvgLatePayment	<p>LatePayments per loan transaction was calculated by deducting provided variables Termdays from DaystoRepayLoan from previous loans dataset using code as provided in (9) above.</p> <p>PREVAvgLatePayment was then derived using derived variables DaystoRepayLoan with PREVFrequency using the same code in (9) above.</p>
11	BirthYear	<p>After converting birthdate column using convert column function SAS Data Studio to date format.</p>  <p>We derive BirthYear as one of our final variables by using “calculated item” and “year” operator in SAS Visual Analytics, to obtain the BirthYear of each customer.</p>  
12	PrevAvgLateFirstPayDate	<p>LateFirstPaymentDate per loan transaction was calculated by deducting provided variables FirstRepaidDate from FirstDueDate from previous loans dataset. Values &gt;= 0 indicates that the loan was paid prior to due date, while values &lt;0 indicate late first payments.</p>



		<pre> 50  data PREVloans2; 51      set work.PREVLOANS1; 52      datemonthyear = datepart(firstduedate_CONVERTED); 53      time = timepart(firstduedate_CONVERTED); 54      format datemonthyear date11. time time.; 55      month = month(datemonthyear); 56      day = day(datemonthyear); 57      year = year(datemonthyear); 58      FirstDueDate = mdy(month, day, year); 59      format FirstDueDate date11.; 60      drop datemonthyear month day year; 61 62      datemonthyear = datepart(firstrepaiddate_CONVERTED); 63      time = timepart(firstrepaiddate_CONVERTED); 64      format datemonthyear date11. time time.; 65      month = month(datemonthyear); 66      day = day(datemonthyear); 67      year = year(datemonthyear); 68      FirstRepaidDate = mdy(month, day, year); 69      format FirstRepaidDate date11.; 70      drop datemonthyear month day year; 71 72      LateFirstPaymentDate = FirstDueDate - FirstRepaidDate; 73  run; </pre> <p>PREVAvgLateFirstPayDate was then derived using derived variables LateFirstPaymentDate with PREVFrequency using code below:</p> <pre> 84  proc sql; 85      create table PREVFreq1 as 86      select distinct CustomerID, PREVFrequency, 87          sum(daystorepayloan) / PREVfrequency as PrevAvgDaystoRepayLoan, 88          sum(Latepayment) / PREVfrequency as PrevAvgLatePayment, 89          sum(LateFirstPaymentDate / PREVfrequency) as PrevAvgLateFirstPayDate 90      from work.PREVFreq 91      group by CustomerID; 92  quit; </pre>
--	--	---

## APPENDIX C: DATA PREPARATION LOG

### Demographics

Item	Data Preparation Issue	Action																																																																													
1	Convert various figures into the correct format in SAS Data Studio	<div>Date columns converted from varchar to datetime format</div> <div>TrainPrevloans:</div> <div><div>1. Convert Column</div><table><thead><tr><th>Source column</th><th>Conversion</th><th>Information or format</th><th>New column</th><th>Length</th><th>Format</th><th>Label</th></tr></thead><tbody><tr><td>approvedate</td><td>datetime</td><td>ANYDTN19</td><td>approvedate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr><tr><td>createndate</td><td>datetime</td><td>ANYDTN19</td><td>createndate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr><tr><td>firstduedate</td><td>datetime</td><td>ANYDTN19</td><td>firstduedate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr><tr><td>firstrepaiddate</td><td>datetime</td><td>ANYDTN19</td><td>firstrepaiddate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr><tr><td>lastrepaiddate</td><td>datetime</td><td>ANYDTN19</td><td>lastrepaiddate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr></tbody></table></div> <div>TrainPerf:</div> <div><div>1. Convert Column</div><table><thead><tr><th>Source column</th><th>Conversion</th><th>Information or format</th><th>New column</th><th>Length</th><th>Format</th><th>Label</th></tr></thead><tbody><tr><td>approvedate</td><td>datetime</td><td>ANYDTN19</td><td>approvedate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr><tr><td>createndate</td><td>datetime</td><td>ANYDTN19</td><td>createndate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr></tbody></table></div> <div>TrainDemographics:</div> <div><div>1. Convert Column</div><table><thead><tr><th>Source column</th><th>Conversion</th><th>Information or format</th><th>New column</th><th>Length</th><th>Format</th><th>Label</th></tr></thead><tbody><tr><td>birthdate</td><td>datetime</td><td>ANYDTN19</td><td>birthdate_CONVERTED</td><td>8</td><td>NLDATE35</td><td></td></tr></tbody></table></div>	Source column	Conversion	Information or format	New column	Length	Format	Label	approvedate	datetime	ANYDTN19	approvedate_CONVERTED	8	NLDATE35		createndate	datetime	ANYDTN19	createndate_CONVERTED	8	NLDATE35		firstduedate	datetime	ANYDTN19	firstduedate_CONVERTED	8	NLDATE35		firstrepaiddate	datetime	ANYDTN19	firstrepaiddate_CONVERTED	8	NLDATE35		lastrepaiddate	datetime	ANYDTN19	lastrepaiddate_CONVERTED	8	NLDATE35		Source column	Conversion	Information or format	New column	Length	Format	Label	approvedate	datetime	ANYDTN19	approvedate_CONVERTED	8	NLDATE35		createndate	datetime	ANYDTN19	createndate_CONVERTED	8	NLDATE35		Source column	Conversion	Information or format	New column	Length	Format	Label	birthdate	datetime	ANYDTN19	birthdate_CONVERTED	8	NLDATE35	
Source column	Conversion	Information or format	New column	Length	Format	Label																																																																									
approvedate	datetime	ANYDTN19	approvedate_CONVERTED	8	NLDATE35																																																																										
createndate	datetime	ANYDTN19	createndate_CONVERTED	8	NLDATE35																																																																										
firstduedate	datetime	ANYDTN19	firstduedate_CONVERTED	8	NLDATE35																																																																										
firstrepaiddate	datetime	ANYDTN19	firstrepaiddate_CONVERTED	8	NLDATE35																																																																										
lastrepaiddate	datetime	ANYDTN19	lastrepaiddate_CONVERTED	8	NLDATE35																																																																										
Source column	Conversion	Information or format	New column	Length	Format	Label																																																																									
approvedate	datetime	ANYDTN19	approvedate_CONVERTED	8	NLDATE35																																																																										
createndate	datetime	ANYDTN19	createndate_CONVERTED	8	NLDATE35																																																																										
Source column	Conversion	Information or format	New column	Length	Format	Label																																																																									
birthdate	datetime	ANYDTN19	birthdate_CONVERTED	8	NLDATE35																																																																										



		<pre> 1 proc sql; 2   title 'Duplicate Rows in Demographics'; 3   select *, count(*) as Count 4   from CASUSER.TrainDemographics_1 5   group by CustomerID 6   having count(*) &gt; 1; 7 run; 8 9 10 proc sql; 11  title 'Duplicate Rows in TrainPrevLoans'; 12  select *, count(*) as Count 13  from CASUSER.TrainPrevLoans 14  group by CustomerID, SystemLoanID 15  having count(*) &gt; 1; 16 run; 17 18 proc sql; 19  title 'Duplicate Rows in TrainPerf'; 20  select *, count(*) as Count 21  from CASUSER.TrainPerf 22  group by CustomerID, SystemLoanID 23  having count(*) &gt; 1; 24 run; </pre> <p>duplicates were removed</p> <p>Duplicate Rows in Demographics</p> <p>Duplicate Rows in TrainPrevLoans</p> <p>Duplicate Rows in TrainPerf</p>																																													
4	We also check for missing data as we will remove variables with missing data > 30% from our analysis	<p><b>TrainPrevLoans:</b> “referredby” has 94.36% missing values and hence will be <b><u>excluded from analysis</u></b></p> <table border="1"> <thead> <tr> <th>referredby</th><th>Frequency</th><th>Percent</th></tr> </thead> <tbody> <tr> <td></td><td>17157</td><td>94.36</td></tr> <tr> <td>Non-missing</td><td>1026</td><td>5.64</td></tr> </tbody> </table> <p><b>TrainPerf:</b> “referredby” has 94.36% missing values and hence will be <b><u>excluded from analysis</u></b></p> <table border="1"> <thead> <tr> <th>referredby</th><th>Frequency</th><th>Percent</th></tr> </thead> <tbody> <tr> <td></td><td>3781</td><td>86.56</td></tr> <tr> <td>Non-missing</td><td>587</td><td>13.44</td></tr> </tbody> </table> <p><b>TrainDemographics:</b></p> <ul style="list-style-type: none"> <li>“bank_branch_clients” has 98.82% of missing values and hence will be <b><u>excluded from analysis</u></b></li> <li>“employment_status_clients” has 14.95% of missing values, but will be <b><u>included</u></b></li> <li>“level_of_education_clients” has 86.48% of missing values and hence will be <b><u>excluded from analysis</u></b></li> </ul> <table border="1"> <thead> <tr> <th>bank_branch_clients</th><th>Frequency</th><th>Percent</th></tr> </thead> <tbody> <tr> <td></td><td>4283</td><td>98.82</td></tr> <tr> <td>Non-missing</td><td>51</td><td>1.18</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th>employment_status_clients</th><th>Frequency</th><th>Percent</th></tr> </thead> <tbody> <tr> <td></td><td>648</td><td>14.95</td></tr> <tr> <td>Non-missing</td><td>3686</td><td>85.05</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th>level_of_education_clients</th><th>Frequency</th><th>Percent</th></tr> </thead> <tbody> <tr> <td></td><td>3748</td><td>86.48</td></tr> <tr> <td>Non-missing</td><td>586</td><td>13.52</td></tr> </tbody> </table>	referredby	Frequency	Percent		17157	94.36	Non-missing	1026	5.64	referredby	Frequency	Percent		3781	86.56	Non-missing	587	13.44	bank_branch_clients	Frequency	Percent		4283	98.82	Non-missing	51	1.18	employment_status_clients	Frequency	Percent		648	14.95	Non-missing	3686	85.05	level_of_education_clients	Frequency	Percent		3748	86.48	Non-missing	586	13.52
referredby	Frequency	Percent																																													
	17157	94.36																																													
Non-missing	1026	5.64																																													
referredby	Frequency	Percent																																													
	3781	86.56																																													
Non-missing	587	13.44																																													
bank_branch_clients	Frequency	Percent																																													
	4283	98.82																																													
Non-missing	51	1.18																																													
employment_status_clients	Frequency	Percent																																													
	648	14.95																																													
Non-missing	3686	85.05																																													
level_of_education_clients	Frequency	Percent																																													
	3748	86.48																																													
Non-missing	586	13.52																																													
5	Convert datetime dates into respective relevant dates and time in SAS Studio	<p><b>TrainPrevLoans:</b> “ApprovedDate”, “CreationDate”, “CloseDate”, “FirstDueDate” and “FirstRepaidDate” are date variables created with code.</p>																																													

		<pre> 14 data PREVloans1; 15 set work.PREVloans; 16 datemonthyear = datepart(approveddate_CONVERTED); 17 time = timepart(approveddate_CONVERTED); 18 format datemonthyear date11. time time.; 19 month = month(datemonthyear); 20 day = day(datemonthyear); 21 year = year(datemonthyear); 22 ApprovedDate = mdy(month, day, year); 23 format ApprovedDate date11.; 24 drop datemonthyear month day year;  25 26 datemonthyear = datepart(creationdate_CONVERTED); 27 time = timepart(creationdate_CONVERTED); 28 format datemonthyear date11. time time.; 29 month = month(datemonthyear); 30 day = day(datemonthyear); 31 year = year(datemonthyear); 32 CreationDate = mdy(month, day, year); 33 format CreationDate date11.; 34 drop datemonthyear month day year;  35 36 datemonthyear = datepart(closeddate_CONVERTED); 37 time = timepart(closeddate_CONVERTED); 38 format datemonthyear date11. time time.; 39 month = month(datemonthyear); 40 day = day(datemonthyear); 41 year = year(datemonthyear); 42 ClosedDate = mdy(month, day, year); 43 format ClosedDate date11.; 44 drop datemonthyear month day year;  45 46 DaystoRepayLoan = ClosedDate - CreationDate; 47 LatePayment = Termdays - DaystoRepayLoan; 48 run; </pre> <pre> 50 data PREVloans2; 51 set work.PREVloans1; 52 datemonthyear = datepart(firstduedate_CONVERTED); 53 time = timepart(firstduedate_CONVERTED); 54 format datemonthyear date11. time time.; 55 month = month(datemonthyear); 56 day = day(datemonthyear); 57 year = year(datemonthyear); 58 FirstDueDate = mdy(month, day, year); 59 format FirstDueDate date11.; 60 drop datemonthyear month day year;  61 62 datemonthyear = datepart(firstrepaiddate_CONVERTED); 63 time = timepart(firstrepaiddate_CONVERTED); 64 format datemonthyear date11. time time.; 65 month = month(datemonthyear); 66 day = day(datemonthyear); 67 year = year(datemonthyear); 68 FirstRepaidDate = mdy(month, day, year); 69 format FirstRepaidDate date11.; 70 drop datemonthyear month day year;  71 72 LateFirstPaymentDate = FirstDueDate - FirstRepaidDate; 73 run; </pre> <p><b>TrainPerf: "ApprovedDate" is a date variable created with code</b></p> <pre> 14 data Existingloans1; 15 set work.ExistingLOANS; 16 datemonthyear = datepart(approveddate_CONVERTED); 17 time = timepart(approveddate_CONVERTED); 18 format datemonthyear date11. time time.; 19 month = month(datemonthyear); 20 day = day(datemonthyear); 21 year = year(datemonthyear); 22 ApprovedDate = mdy(month, day, year); 23 format ApprovedDate date11.; 24 run; </pre> <p><b>TrainDemo: "BirthDate" is a date variable created with code</b></p> <pre> 10 data demographics1; 11 set work.demographics; 12 datemonthyear = datepart(birthdate_CONVERTED); 13 time = timepart(birthdate_CONVERTED); 14 format datemonthyear date11. time time.; 15 month = month(datemonthyear); 16 day = day(datemonthyear); 17 year = year(datemonthyear); 18 Birthdate = mdy(month, day, year); 19 format Birthdate date11.; 20 drop datemonthyear month day year time; 21 run; </pre>
6	Calculation of Frequency from RFM Variables	<p>To obtain frequency, we count the number of time in the converted datasets for each customer to get the respective frequencies.</p> <p><b>TrainPrevLoans:</b></p> <pre> 75 proc sql; 76 create table PREVFreq as 77 select CustomerID, max(loannumber) as MaxLoanNumber, 78 DaystoRepayLoan, LatePayment, LateFirstPaymentDate, 79 count(distinct Time) as PREVFrequency 80 from work.PREVloans2 81 group by CustomerID; 82 quit;  83 84 proc sql; 85 create table PREVFreq1 as 86 select distinct CustomerID, PREVFrequency, 87 sum(daystorepayloan) / PREVfrequency as PrevAvgDaystoRepayLoan, 88 sum(Latepayment) / PREVfrequency as PrevAvgLatePayment, 89 sum(LateFirstPaymentDate / PREVfrequency) as PrevAvgLateFirstPayDate 90 from work.PREVFreq 91 group by CustomerID; 92 quit; </pre> <p>In TrainPrevLoans, we attempted to derive frequency from finding out maximum loannumber as well, but realized that loannumber did not accurately capture the frequency of previous loans by each customer. MaxLoanNumber increased the frequencies of loan taken by each customer and hence to be conservative, we use frequency based on count of unique</p>

		<div>transactions by each customer as reflected in the dataset.</div> <div><table><thead><tr><th></th><th>customerid</th><th>MaxLoanNumber</th><th>PREVFrequency</th></tr></thead><tbody><tr><td>76</td><td>8a399f284741b63b0147496a04db1eb4</td><td>3</td><td>1</td></tr><tr><td>77</td><td>8a688b474ed5e088014ed8b6c50c5f51</td><td>4</td><td>2</td></tr><tr><td>78</td><td>8a688b474ed5e088014ed8b6c50c5f51</td><td>4</td><td>2</td></tr><tr><td>79</td><td>8a6a96044539adf801453c576ed213c4</td><td>15</td><td>15</td></tr><tr><td>80</td><td>8a6a96044539adf801453c576ed213c4</td><td>15</td><td>15</td></tr><tr><td>81</td><td>8a6a96044539adf801453c576ed213c4</td><td>15</td><td>15</td></tr></tbody></table></div> <div>TrainPerf:<div><pre>26 proc sql; 27 create table Freq as 28     select CustomerID, max(loannumber) as MaxLoanNumber, 29     count(distinct Time) as Frequency 30     from work.ExistingLoans1 31     group by CustomerID; 32 quit;</pre></div><div>Similar to TrainPrevLoans, there were also discrepancies between maxloannumber. We will not use frequency derived from TrainPerf as frequencies were identified as 1 for all customers, indicating that each customer only had 1 existing loan.</div><div><table><thead><tr><th></th><th>customerid</th><th>MaxLoanNumber</th><th>Frequency</th></tr></thead><tbody><tr><td>1</td><td>8a1088a0484472eb01484669e3ce4e0b</td><td>2</td><td>1</td></tr><tr><td>2</td><td>8a1a1e7e4f07f8b014f797718316cad</td><td>5</td><td>1</td></tr><tr><td>3</td><td>8a1a32fc49b632520149c3b8df85139</td><td>8</td><td>1</td></tr></tbody></table></div></div>		customerid	MaxLoanNumber	PREVFrequency	76	8a399f284741b63b0147496a04db1eb4	3	1	77	8a688b474ed5e088014ed8b6c50c5f51	4	2	78	8a688b474ed5e088014ed8b6c50c5f51	4	2	79	8a6a96044539adf801453c576ed213c4	15	15	80	8a6a96044539adf801453c576ed213c4	15	15	81	8a6a96044539adf801453c576ed213c4	15	15		customerid	MaxLoanNumber	Frequency	1	8a1088a0484472eb01484669e3ce4e0b	2	1	2	8a1a1e7e4f07f8b014f797718316cad	5	1	3	8a1a32fc49b632520149c3b8df85139	8	1
	customerid	MaxLoanNumber	PREVFrequency																																											
76	8a399f284741b63b0147496a04db1eb4	3	1																																											
77	8a688b474ed5e088014ed8b6c50c5f51	4	2																																											
78	8a688b474ed5e088014ed8b6c50c5f51	4	2																																											
79	8a6a96044539adf801453c576ed213c4	15	15																																											
80	8a6a96044539adf801453c576ed213c4	15	15																																											
81	8a6a96044539adf801453c576ed213c4	15	15																																											
	customerid	MaxLoanNumber	Frequency																																											
1	8a1088a0484472eb01484669e3ce4e0b	2	1																																											
2	8a1a1e7e4f07f8b014f797718316cad	5	1																																											
3	8a1a32fc49b632520149c3b8df85139	8	1																																											
7	Calculation of Recency from RFM Variables	<div>Using last date numerical value, we add 1 to the last date to arrive at figures to be used to derive recency.</div> <div>TrainPrevLoans: 21028 was the last date and 21029 was the value used to deduct PREVLastDate</div> <div><pre>104 data PREVRM1; 105 set PREVRM; 106 PREVRecency = 21029 - PREVLastDate; 107 format PREVLastDate date11.; 108 format PREVFirstApprovedDate date11.; 109 run;</pre></div> <div>TrainPerf: 21031 was the last date and 21030 was the value used to deduct ExistingApprovedDate</div> <div><pre>43 data RM1; 44 set RM; 45 Recency = 21031 - ExistingApprovedDate; 46 format ExistingApprovedDate date11.; 47 run;</pre></div>																																												
8	Calculation of Average (Monetary) Variables used for further analysis	<div>TrainPrevLoans: New Variables “PREVAvgDaystoRepayLoans”, “PREVAvgLatePayment” and “PREVAvgLateFirstPayDate” derived using PREVFrequency calculated.</div>																																												

		<pre> 84 proc sql; 85 create table PREVFreq1 as 86     select distinct CustomerID, PREVFrequency, 87         sum(daystorepayloan) / PREVfrequency as PrevAvgDaystoRepayLoan, 88         sum(Latepayment) / PREVfrequency as PrevAvgLatePayment, 89         sum(LateFirstPaymentDate / PREVfrequency) as PrevAvgLateFirstPayDate 90     from work.PREVFreq 91     group by CustomerID; 92 quit; 93 94 proc sql; 95 create table PREVRM as 96     select CustomerID, sum(LoanAmount) as LoanAmount, sum(totaldue) as Totaldue, 97         max('AnnualInterest'n) as AnnualInterest, 98         max('ApprovedDate'n) as PREVLastDate, 99         min('ApprovedDate'n) as PREVFirstApprovedDate 100     from work.PREVRloans1 101     group by CustomerID; 102 quit; </pre> <p>TrainPerf: New Variables “AverageLoanAmount”, “AverageTotalDue” and “AverageInterest” derived using Frequency calculated.</p> <pre> 49 proc sql; 50 create table RFM as 51     select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, Recency, 52         Frequency, ExistingApprovedDate, 53         LoanAmount/Frequency as AverageLoanAmount, 54         TotalDue/Frequency as AverageTotalDue, 55         AnnualInterest/Frequency as AverageInterest 56     from RM1 as a left join Freq as b 57         on a.CustomerID = b.CustomerID; 58 quit; </pre>
9	Combine relevant columns of variables	<p><b>TrainPrevLoans:</b> We combined PREVFreq1 table with PREVRM1 to derive <b><u>PREVRFM</u></b> for combining with TrainDemographics.</p> <pre> 110 proc sql; 111 create table PREVRFM as 112     select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, PREVRecency, 113         PREVFrequency, PREVFirstApprovedDate, 114         LoanAmount/PREVFrequency as AveragePrevLoanAmount, 115         TotalDue/PREVFrequency as AveragePrevTotalDue, 116         AnnualInterest/PREVFrequency as AveragePrevInterest, 117         PrevAvgDaystoRepayLoan, 118         PrevAvgLatePayment, 119         PrevAvgLateFirstPayDate 120     from PREVRM1 as a left join PREVFreq1 as b 121         on a.CustomerID = b.CustomerID; 122 quit; --- </pre> <p><b>TrainPrevLoans:</b> We combined Freq table with RM tables to derive RFM, then combine RFM with ExistingLoans to include good_bad_flag binary outcome predictors to derive <b><u>RFM2</u></b> prior to combining with TrainDemographics.</p> <pre> 49 proc sql; 50 create table RFM as 51     select a.CustomerID, LoanAmount, TotalDue, AnnualInterest, Recency, 52         Frequency, ExistingApprovedDate, 53         LoanAmount/Frequency as AverageLoanAmount, 54         TotalDue/Frequency as AverageTotalDue, 55         AnnualInterest/Frequency as AverageInterest 56     from RM1 as a left join Freq as b 57         on a.CustomerID = b.CustomerID; 58 quit; 59 60 proc sql; 61 create table RFM2 as 62     select a.CustomerID, AverageLoanAmount, AverageTotalDue, AverageInterest, Recency, 63         Frequency, ExistingApprovedDate, good_bad_flag 64     from RFM as a left join Existingloans as b 65         on a.CustomerID = b.CustomerID; 66 quit; </pre>
10	Combine PREVRFM and RFM2 with	<p>We combined PREVRFM with TrainDemographics and RFM2 with TrainDemographics with left join with left join using customerID as a common</p>

## TrainDemographics

field – to obtain both PL\_Demo\_1 and EL\_Demo\_1 respectively.

```
23 proc sql;
24 create table PL_Demo_1 as
25 select a.CustomerID, birthdate, prevfirstapproveddate, bank_account_type, longitude_gps, latitude_gps,
26 bank_name_clients, bank_branch_clients, employment_status_clients, level_of_education_clients,
27 averageprevloanamount, averageprevtotaldue, averageprevinterest, PREVfrequency, PREVrecency,
28 PREVavgDaystoRepayLoan, PREVavgLatePayment, PREVavgLateFirstPayDate
29 from PREVRFM as a left join demographics1 as b /*use left we get all data*/
30 on a.CustomerID = b.CustomerID;
31 quit;
32
33 proc sql;
34 create table EL_Demo_1 as
35 select a.CustomerID, birthdate, existingapproveddate, bank_account_type, longitude_gps, latitude_gps,
36 bank_name_clients, bank_branch_clients, employment_status_clients, level_of_education_clients,
37 averageloanamount, averagetotaldue, averageinterest, frequency, recency, good_bad_flag
38 from RFM2 as a left join demographics1 as b /*use left we get all data*/
39 on a.CustomerID = b.CustomerID;
40 quit;
```

PL\_Demo\_1 and EL\_Demo\_1 were then combined using inner join to using customerID as a common field to eliminate 9 customers who are new-to-business customers as they do not have previous loan records, which is more relevant as a predictor of good\_bad\_flag when customers create a request for a loan.

```
42 proc sql;
43 create table EL_PL_Demo_1 as
44 select a.CustomerID, a.birthdate, a.existingapproveddate, a.bank_account_type, a.longitude_gps, a.latitude_gps,
45 a.bank_name_clients, a.bank_branch_clients, a.employment_status_clients, a.level_of_education_clients,
46 b.CustomerID, b.birthdate, b.prevfirstapproveddate, b.bank_account_type, b.longitude_gps, b.latitude_gps,
47 b.bank_name_clients, b.bank_branch_clients, b.employment_status_clients, b.level_of_education_clients,
48 averageprevloanamount, averageprevtotaldue, averageprevinterest, PREVfrequency, PREVrecency,
49 PREVavgDaystoRepayLoan, PREVavgLatePayment, PREVavgLateFirstPayDate,
50 averageloanamount, averagetotaldue, averageinterest, a.frequency, a.recency, good_bad_flag
51 from EL_Demo_1 as a inner join PL_Demo_1 as b /*use left we get all data*/
52 on a.CustomerID = b.CustomerID;
53 quit;
```

Following which, we export data derived using SAS Studio into csv

« EL\_PL\_DEMO\_1 Table rows: 4359 Columns: 25 of 25 Rows 1 to 200

	customerid	Birthdate
1	8a1088a0484472eb01484669e3ce4e0b	09-NOV-1989
2	8a1a1e7e4f707f8b014f797718316cad	18-OCT-1979
3	8a1a32fc49b632520149c3b8fd85139	29-JAN-1979
4	8a1eb5ba49a682300149c3c068b806c7	25-NOV-1978
5	8a1edbf14734127f0147356fdb1b1eb2	.
6	8a26bd845089f1d7015090b1d6f53bad	03-APR-1987
7	8a2a81a74ce8c05d014cfb32a0da1049	15-JAN-1972
8	8a2ac4745091002b0150a144bcb58b7	.
9	8a2ad9ce4c453e06014c4b3175e52407	12-SEP-1974
10	8a33a06e4a5075c2014a5295aa0c2224	30-APR-1977
11	8a3442825124396701513514dad40fa7	07-AUG-1986
12	8a390a2150ad97330150aebdd8ef7456	.
13	8a390a2150ad97330150aebdd8ef7456	.

Column selection

- Add to My Favorites
- Generate code
- Open in a query
- Open in a task
- Export
- Show labels
- ✓ Show names
- ✓ Show filter
- Show distinct rows
- Row paging
- Print
- Table properties

Folder Shortcuts

- Viya-Share
- SAS Content
  - Conversational Flows
  - ISS602
    - valerianayap.2023
      - Assign1
      - Assign2
      - DAL\_G1\_Group Project
      - Hands-on Exercise 1
      - Hands-on Exercise 2
      - Hands-on Exercise 3
      - Hands-on Exercise 4
      - Hands-on Exercise 5

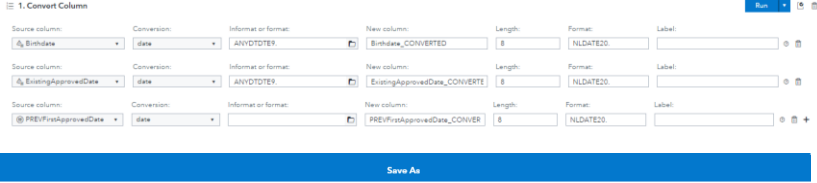
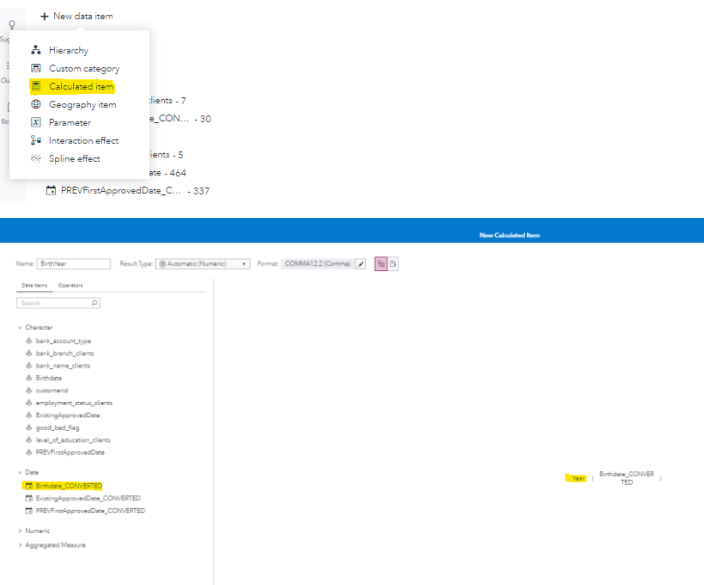
EL\_PL\_DEMO\_1.csv

EL\_PL\_DEMO\_2.csv

EL\_PL\_DEMO\_3.csv

EL\_PL\_DEMO\_4.csv

Name: EL\_PL\_DEMO\_4.csv Type: Comma-delimited (\*.csv)

11	Further handle format of imported data file prior to	<p>EL_PL_Demo_4.csv is then imported using SAS Data Explorer – Manage Data and saved as EL_PL_Demo_4 table. The table is then prepared in SAS Data Studio to convert date variables from varchar to date format.</p>  <p>1. Convert Column</p> <p>Source column: Birthdate Conversion: date Informat or format: ANYDTDTY New column: Birthdate_CONVERTED Length: 8 Format: NDATE20 Label: </p> <p>Source column: ExistingApprovedDate Conversion: date Informat or format: ANYDTDTY New column: ExistingApprovedDate_CONVERTED Length: 8 Format: NDATE20 Label: </p> <p>Source column: PREVFirstApprovedDate Conversion: date Informat or format: ANYDTDTY New column: PREVFirstApprovedDate_CONVERTED Length: 8 Format: NDATE20 Label: </p> <p>Save As</p> <p>Search</p> <p>Folders &gt; SAS Content &gt; ISS5602 &gt; valerianyp.2023 &gt; DAL_G1_Group Project</p> <p>Assign1 EL_PL_DEMO_1_1 Assign2 EL_PL_DEMO_2_1 DAL_G1_Group Project TRAINDEMOGRAPHICS_1 Hands-on Exercise 1 TRAINPERF_1 Hands-on Exercise 2 TRAINPERF_DEMO_JOIN_1 Hands-on Exercise 3 TRAINPREVLOANS_1 Hands-on Exercise 4</p> <p>Name: EL_PL_DEMO_2_1 Type: Data plan</p> <p><input checked="" type="radio"/> Save plan and target table <input type="radio"/> Save plan <input type="radio"/> Save target table</p> <p>Target table name: EL_PL_DEMO_2_1 Label: Enter label Format: @ Library: cas-shared-default/CASUSER/valerianyp...</p> <p><input type="checkbox"/> Save as an in-memory table only</p> <p>If the name of the target table already exists: <input type="radio"/> Cancel save <input checked="" type="radio"/> Replace table</p> <p>Save Cancel</p>
12	To Derive New Variable – “BirthYear”	<p>We derive BirthYear as one of our final variables by using “calculated item” and “year” operator in SAS Visual Analytics, to obtain the BirthYear of each customer.</p>  <p>+ New data item</p> <p>Hierarchy Custom category Calculated item Geography item Parameter Interaction effect Spline effect</p> <p>PREVFirstApprovedDate_C... - 337</p> <p>New Calculated Item</p> <p>Name: BirthYear Result Type: Automatic(Numeric) Format: COMMA12.2(Gamma) </p> <p>Data items: Birthdate_CONVERTED, ExistingApprovedDate_CONVERTED, PREVFirstApprovedDate_CONVERTED</p> <p>Operations: YEAR(Birthdate_CONVERTED)</p>