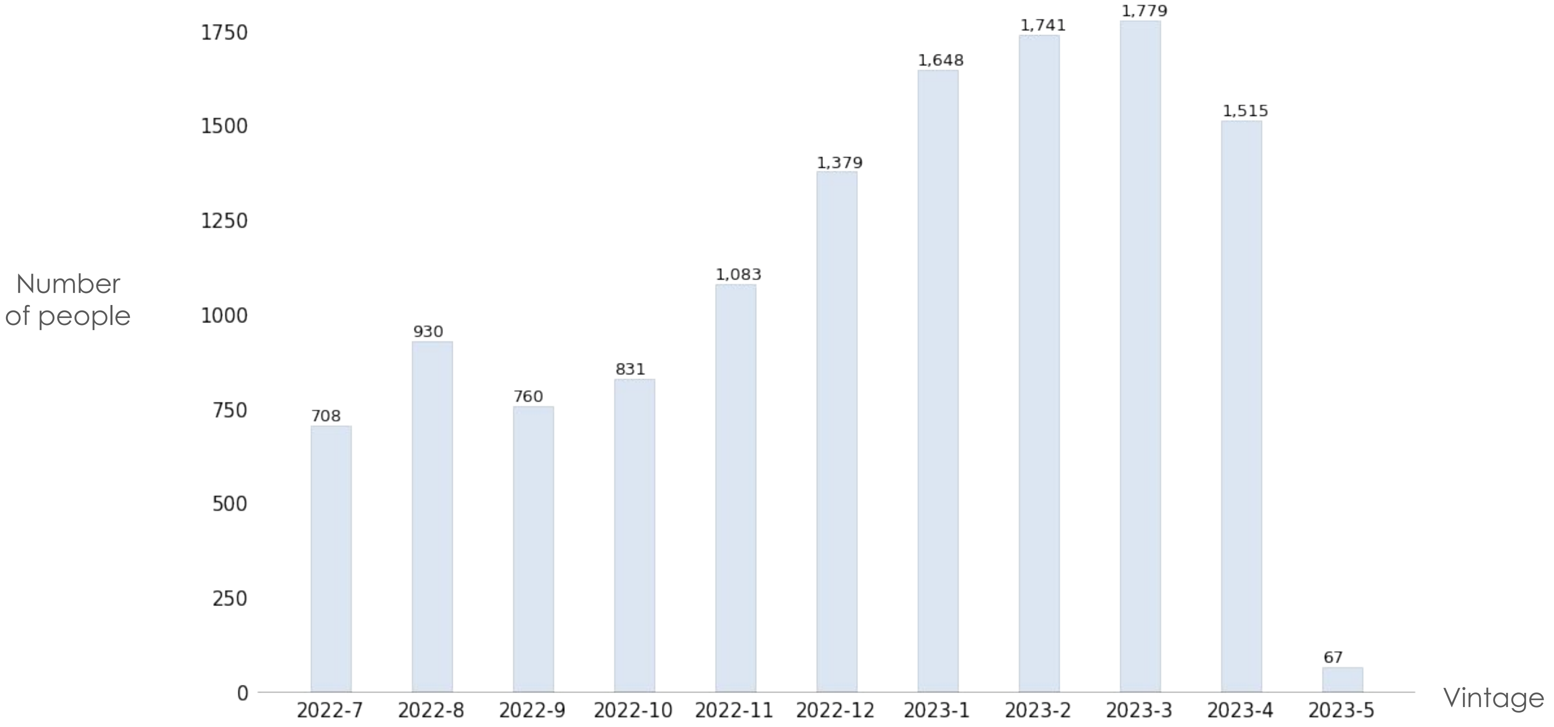
Risk Model Challenge

Loan to buy a smartphone

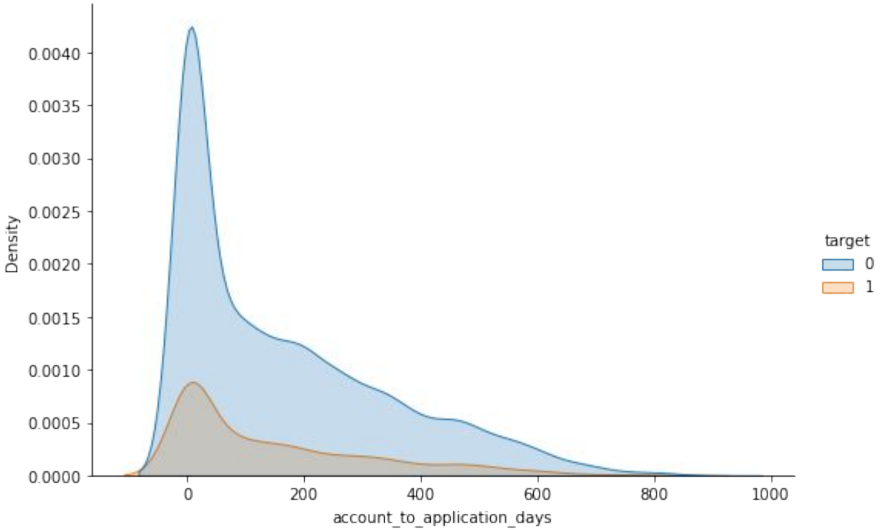
Highlights

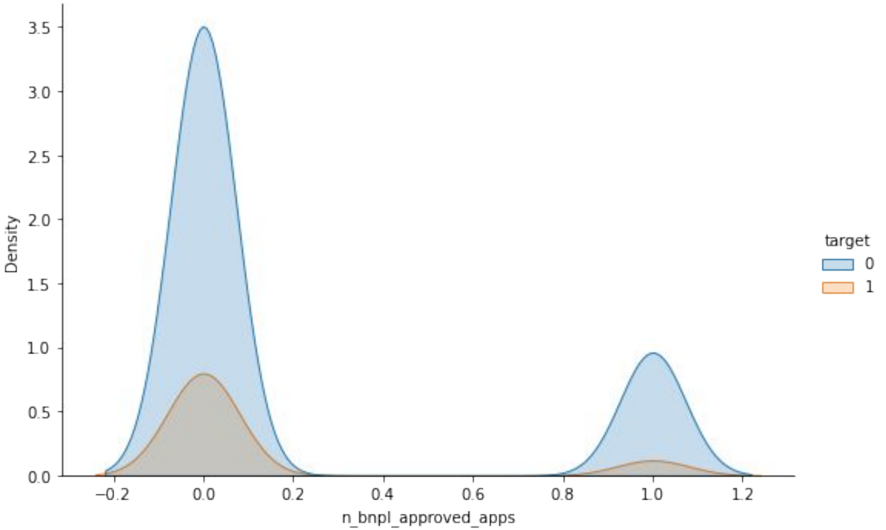
- There are 14,454 people with information for training the mode
  - 2,700 defaulters (19%)
  - 11,754 good payers (81%)
- There are people from July 2022 to May 2023



Highlights

- Bad customers tend to apply more frequently for smartphone financing and BNPL products, but they are approved less often than good customers. Additionally, they receive more inquiries from external entities





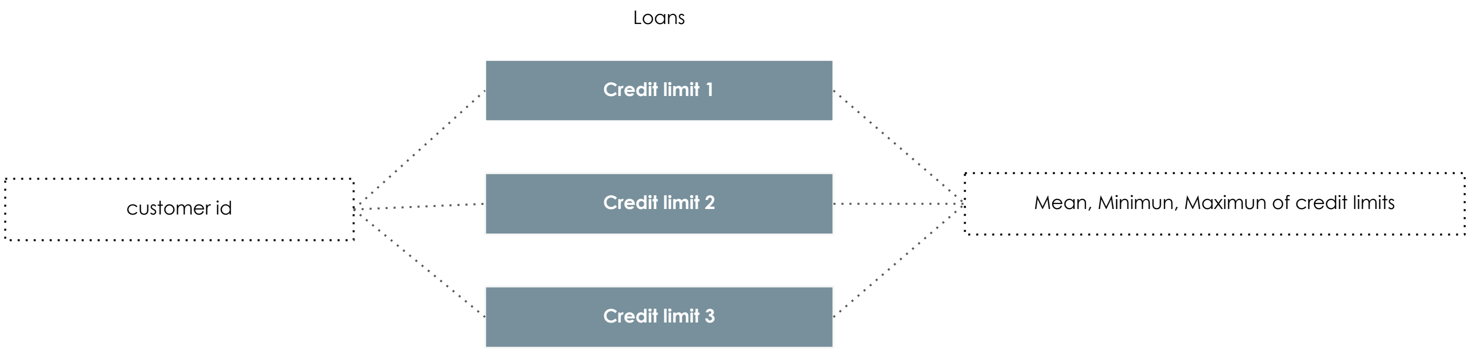
Highlights

- Let's see which variables have the highest percentage of null values in the main dataset

Variable	Porcentaje de nulos
customer_id	0%
loan_id	0%
ACC_CREATION_DATETIME	0%
APPLICATION_DATETIME	0%
LOAN_ORIGINATION_DATETIME	0%
max_days_late	0%
target	0%
account_to_application_days	0%
n_sf_apps	53%
first_app_date	53%
last_app_date	53%
n_bnpl_apps	40%
n_bnpl_approved_apps	40%
first_bnpl_app_date	40%
last_bnpl_app_date	40%
n_inquiries_l3m	37%
n_inquiries_l6m	37%

Feature engineering

- To create variables at the customer_id level from credit reports, the following steps were taken, represented by this diagram. This way, we added the information of each customer's accounts at the customer level



Feature engineering

- To process the categorical variables at the customer id level, the following was done:
  - One Hot Encoding in credit reports
  - Aggregate to customer id level

Step 1

customer id	Feature
1	category 1
1	category 2
1	category 3

Step 2

customer id	Feature category 1	Feature category 2	Feature category 3
1	1	0	0
1	0	1	0
1	0	0	1

Step 3

customer id	sum_Feature category 1	sum_Feature category 2	sum_Feature category 3
1	1	1	1

customer id	max_Feature category 1	max_Feature category 2	max_Feature category 3
1	1	1	1

Preprocessing

- The following preprocessing steps were applied to the data to have it ready for training
  - Zero imputer: Replace nulls with 0's, as all variables make sense to replace them with 0
  - Max winsorizer: Cap values that are more distant from the mean by three standard deviations with this method to remove outliers

Data

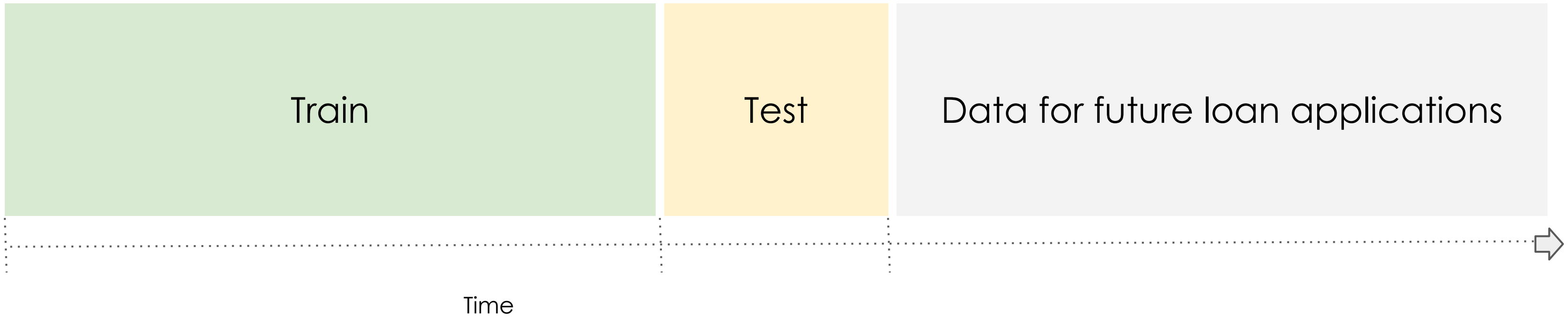
Zero Imputer

Max winsorizer

Preprocessed data

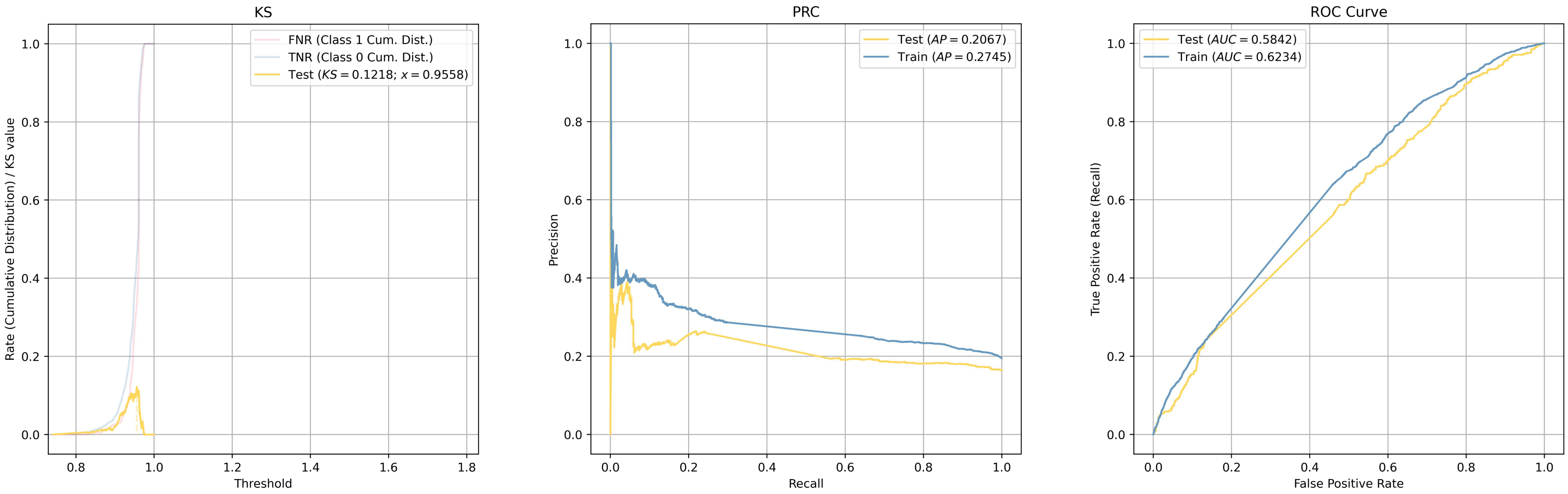
Train/test split

- Since we are working with a model that will be used in the future with data we haven't seen, it is preferable to split the training and test sets based on time. This way, we achieve a better evaluation compared to using a random partition where the time variable is excluded from the split



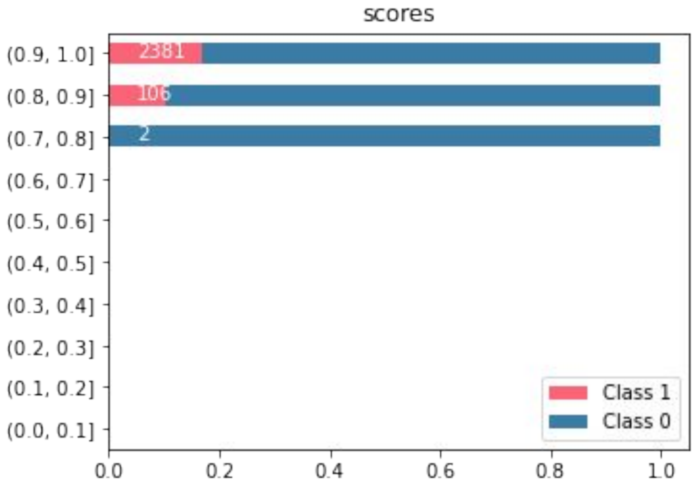
Training and evaluation

- We used an xgboost model and RandomizedSearchCV to find the best parameters. Since we are working with an imbalanced dataset, special attention was given to the scale_pos_weight parameter, which aims to correct this imbalance.



Training and evaluation

- As you can see, the performance is not spectacular; however, it is the best that could be achieved with the current data. Efforts were made to minimize overfitting. It appears that the model is good at distinguishing between payers and non-payers, as the number of non-payers increases as the probability of non-payment rises

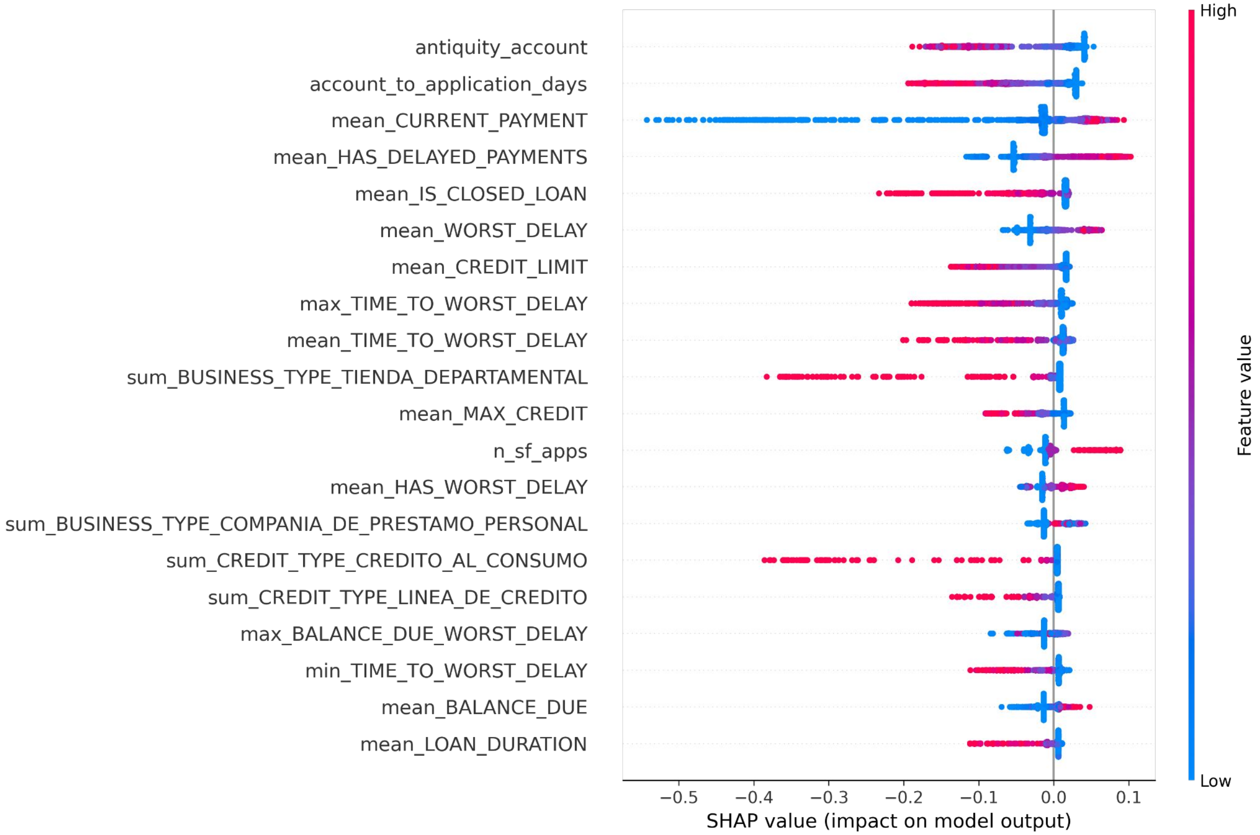


- 'roc_auc': 0.584
- 'threshold': 0.951
- 'f1_score': 0.294
- 'precision': 0.185
- 'recall': 0.711

It seems that with a threshold of 0.951, we have a very low precision but an acceptable recall

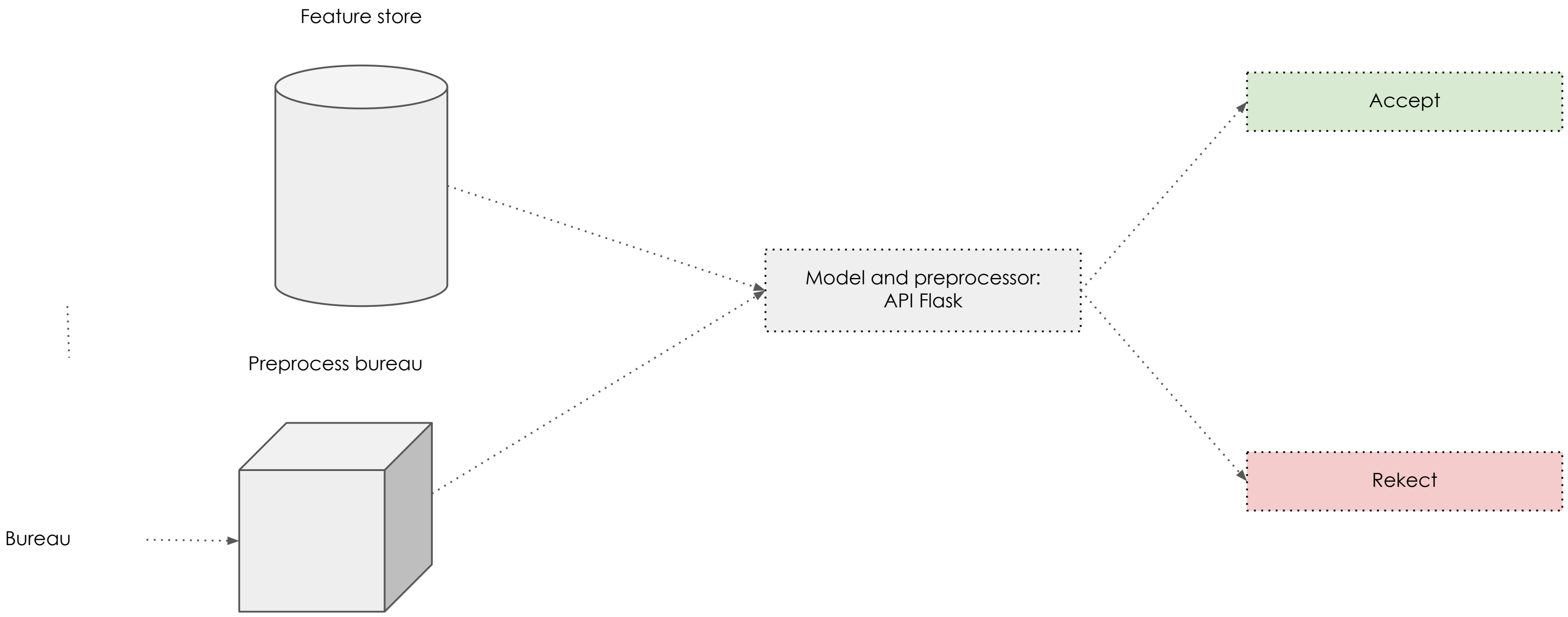
Shap values

- Top most important variables in the model



Model usage

- We can use the model through a Flask API to score each credit applicant in real time. However, for this, we need the following:
  - Feature store to store variables from Bankaya's own sources
  - API to process variables coming from external sources such as credit bureaus



Interest rate

- To assign a line of credit to each client, we can base it on their probability of non-payment as follows:

Probability of default	Weight
[0, 0.12)	Weight_1
[0.12, 0.24)	Weight_2
…	…
…	…
…	…
[0.88,1]	Weight_n

Interest rate = Weight_i * reference rate