Methodology Note

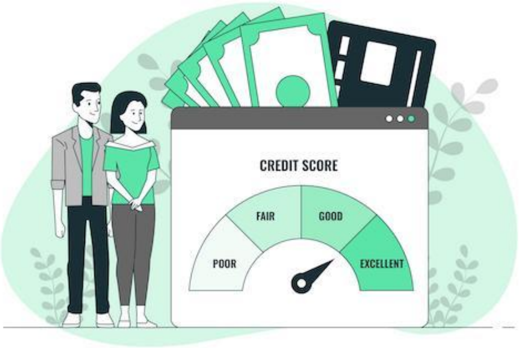
1. Introduction

Many people experience difficulties in obtaining loans if they don’t have a sufficient credit history or when they apply for the first time and therefore don’t have a credit history at all.

Unfortunately, these people are exposed to a high risk of coming across unreliable lenders. **Prêt à dépenser** aims to provide a better financial inclusion for this population and to improve their borrowing experience. In order to compensate for the lack of credit history data, **Prêt à dépenser** takes advantage of a variety of alternative data such as:

- personal
- behavioral
- communication
- demographic
- transactional

etc, that allow to anticipate their clients' payment abilities.



Storyset image on Freepik

Prêt à dépenser aims to use machine learning techniques to issue the predictions and calculate the client score. The present note will introduce, explain and assess different classification algorithms, evaluation metrics and optimizations, a Data Drift analysis and further possible improvements.

2. Model training lifecycle

Training machine learning models is an iterative, multidirectional process composed of the following steps:

1. Define the business problem and outcome.
2. Collect and prepare the data.
3. Train a model with a selected algorithm.
4. Define an evaluation metric and assess the model.
5. Tune model hyperparameters.

2.1 Business problem and outcome

The business objective consists in predicting future payment behavior of clients by calculating the client score going from 0 to 1, where 0 defines non-risky clients and 1 stands for clients with payment difficulties.

2.2 Data

2.2.1 Origin

The data used to train the models comes from Home Credit Kaggle Competition and consists of multiple csv files with clients personal information, application details, previous credits and credit card balance history. The columns' description is also provided.

2.2.2 Feature engineering

The feature engineering was performed based on the Kaggle notebook that obtained a high score on the Kaggle competition and presents an interesting feature transformation from the business point of view.

The following transformations have been performed:

- combine datasets
- aggregate features per client id
- apply different set of functions to the aggregated features (`mean`, `sum`, `min`, `max`, `var`) based on the example of the Kaggle notebook.

2.2.3 Data preprocessing

The model training pipeline was configured in a way to allow different combinations of preprocessing techniques depending on the machine learning model in order to find the best setup for each model. Overall, the following methods were applied and evaluated:

- Encoding of the categorical features using OneHotEncoder.
- Scaling of continuous features with StandardScaler, MinMaxScaler, RobustScaler.
- Normalization of the distribution for continuous features with PowerTransformer.
- Filling missing values (median value for numerical features and most frequent for categorical)
- Capping outliers using IQR method.

2.2.4 Data optimization

In addition to preprocessing, the following optimizations were applied to the data in order to reduce the computational cost of modeling and improve the performance of the model:

- downcast data types that reduced memory usage by 47%.
- select 25 most relevant features by removing correlated (>=0.7) and low variance variables (with 0.1 variance threshold) and applying machine learning methods (using *coef_* and *features_importances_* model apis).

2.2.5 Preparing data for training

It is important to have a robust way to evaluate the performance of the model, therefore the data was split in training (80%) and test sets (20%). The training set was used during the model training, and the test set was used for the model evaluation.

2.3 Model training

The next step was to choose an algorithm depending on the objective and the type of data. We have a binary classification problem and labeled data, therefore we used supervised learning algorithms suitable for categorizing observations into one of two classes. Each of the following models (except for the Dummy Classifier) was trained with a different combination of preprocessing techniques and engineered features in order to evaluate their relevance.

2.3.1 Dummy Classifier

Dummy Classifier serves as a simple baseline to compare against more complex classifiers, it makes predictions that ignore the input features and returns the most frequent class label in our case.

2.3.2 Logistic Regression

Logistic regression is a statistical method for binary classification. It models the relationship between a dependent binary variable and independent variables by estimating the probability of the binary outcome. It uses a logistic function to transform linear combinations of the independent variables, making it suitable for predicting probabilities and classifying data into two categories based on a threshold.

2.3.3 Random Forest Classifier

Random forest is an ensemble learning algorithm that uses a collection of decision trees to make predictions. Each decision tree is trained on a different subset of the data, and the predictions of all the trees are averaged to produce the final prediction. This makes random forest very robust to overfitting and able to handle complex relationships between the features and the target variable.

2.3.4 LightGBM

LightGBM is a highly efficient gradient boosting framework that uses tree based learning algorithms. Gradient Boosting refers to a methodology in machine learning where an ensemble of weak learners is used to improve the model performance in terms of efficiency, accuracy, and interpretability.

2.3.5 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It's a parallelized and carefully optimized version of the gradient boosting algorithm.

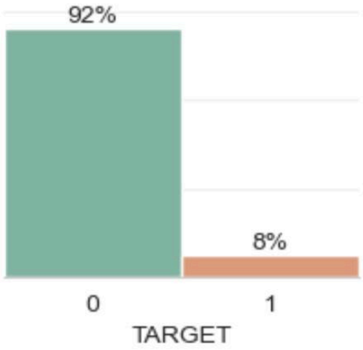
2.3.6 Threshold moving

The default threshold for interpreting probabilities to class labels is 0.5, however, for the data with a high class imbalance, the default threshold can result in poor performance. In order to find the optimal threshold the **Youden's J statistic** with the following formula was used:

J = max(Sensitivity + Specificity − 1)

2.3.7 Class imbalance

The dataset presents a severe class imbalance:



The class 0 has 92% of observations while class 1 only 8%. This can have a significant impact on the performance of the model that will tend to be biased towards the majority class and ignore the minority class. Therefore, regularization is important to prevent the overfitting of the majority class.

2 techniques were explored to address the class imbalance issue:

- **Oversampling** the minority class in the training dataset prior to fitting a model with SMOTE and SMOTE-NC techniques. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. SMOTE-NC is a version of SMOTE adapted for datasets containing both numerical and categorical features.
- **Class weights**: assign higher weights to the samples of the minority class and lower weights to the majority class during the training process, this will allow the model to reduce bias towards the majority class.

2.4 Evaluation

In order to assess the model's performance and establish benchmarks we need to determine the most relevant evaluation metrics for our use case. Every business problem is different and should be optimized differently.

In estimating credit risk models, we have 2 Error Types, which affect the profitability and credit losses of the company:

- **Type I Error, False Positives**: customers predicted as potential defaulters but did not default, they affect profitability as they represent potential clients that are lost.

- **Type II Error, False Negatives**: customers predicted as not defaulter and who actually defaulted, they can cause an important money loss for the company.

It is generally believed that the cost of incorrectly classifying a bad customer as a good customer is much higher and causes a more serious damage to credit institutions, therefore the focus was made on the **Type II Error**. Some researchers reported that the ratio of misclassification costs is 5:1. However, the use of this relative cost ratio can be extended, in terms of sensitivity analysis, to higher cost ratios (i.e. 7:1, 10:1 etc) as noted by [1][2]. We implemented the recommended ratio of 10:1 for this project.

The model was evaluated on the test set with the following metrics:

- **Ratio of 10*FN + FP**
  False Negatives are given more weight as their number should be minimized.

- **Fβ-score with β = 10**
  The Fβ-score is the general form of the F-score where β allows for an unequal weighting of Precision and Recall. In our case more weight is given to Recall.

- **ROC AUC**
  Area under the ROC Curve, measures the entire two-dimensional area underneath the ROC curve which is a graph that plots True Positive Rate vs. False Positive Rate at different classification thresholds.

- **Recall**
  Percentage of samples that were correctly identified as belonging to the positive class.

- **Accuracy**
  Proportion of observations that were correctly predicted.

2.5 Fine-tuning

Hyperparameter optimization is the process of searching for the most accurate hyperparameters for the given algorithm and is required to get the best performance of the model.

After splitting the training set in stratified folds, the following approaches were used to find the best combination of hyperparameters for each model:

- **GridSearchCV**
  This method exhaustively considers all parameter combinations.

- **RandomizedSearchCV**
  Implements a randomized search over parameters, where each setting is sampled from a distribution over possible parameter values.

- **HyperOpt**
  A Bayesian Optimization method meaning it is intelligently learning which combinations of parameters work well at every iteration. This method was used only for LightGBM and XGBoost models due to a high number of parameters

3. Model training results

	Dummy	LightGBM	XGBoost	Random Forest	Logistic Regression
Cost Ratio	0.79	0.56	0.59	0.6	0.61
Fβ-score	0	0.64	0.65	0.61	0.61
AUC	0.5	0.73	0.71	0.7	0.69
Recall	0	0.66	0.68	0.64	0.63
Accuracy	0.92	0.68	0.64	0.66	0.65
Duration	7.8s	1.9min	4.5min	2.4min	1.7min

The **LightGBM** model performed the best at Cost Ratio Score, AUC and Accuracy. XGBoost gave a good Fβ-score and Recall, but it was the most time consuming.

The following preprocessing configuration for LIghtGBM gave the best results:

Data preprocessing:

Filling missing values	No
Scaling (MinMaxScaler)	Yes
Normalizing distribution	No
Handling outliers	No
Encoding categorical (OneHotEncoding)	Yes

LightGBM is able to handle missing values without imputation, by using special values to represent missing data in the tree. It is also robust to outliers.

Imbalanced classes configuration:
The model's `class_weight` parameter with `balanced` value performed better than oversampling with SMOTE and SMOTE-NC

Hyperparameters:

n_estimators	158
num_leaves	111
max_depth	4
learning_rate	0.09
min_data_in_leaf	135
subsample_for_bin	20000
colsample_bytree	0.59
lambda_l1	0.71
lambda_l2	0.19

Optimal threshold: 0.5

4. Model interpretation

Feature importance refers to a class of techniques for assigning scores to input features to a model that indicates the relative importance of each feature when making a prediction. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable.

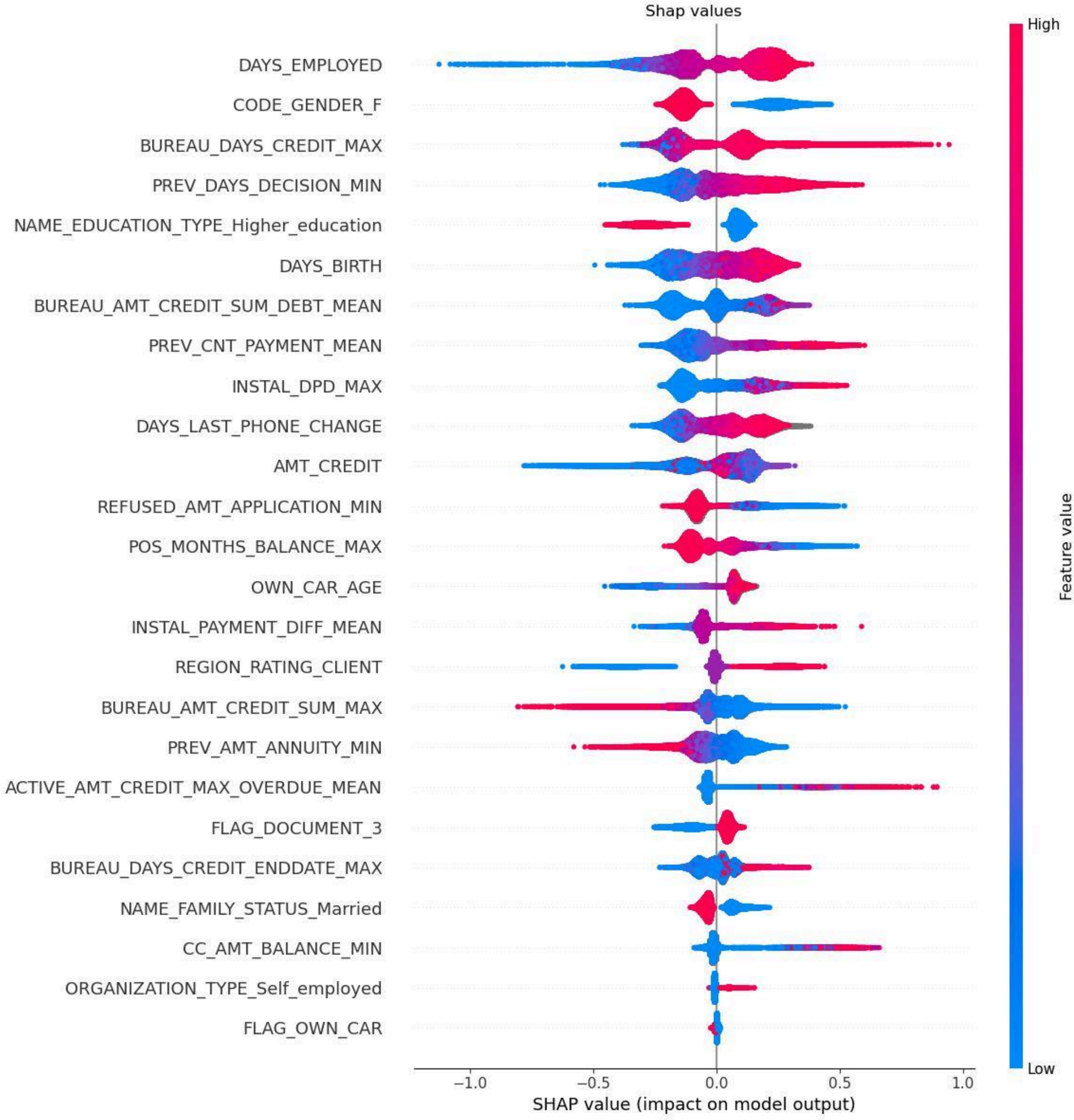
There are different methods to calculate feature importance, in this project we focused on:

- **Feature importance** property of lightGBM model
- **SHAP** library

The results will be presented using the SHAP library. SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. The goal of SHAP is to explain the prediction of an instance x by computing the contribution of each feature to the prediction.

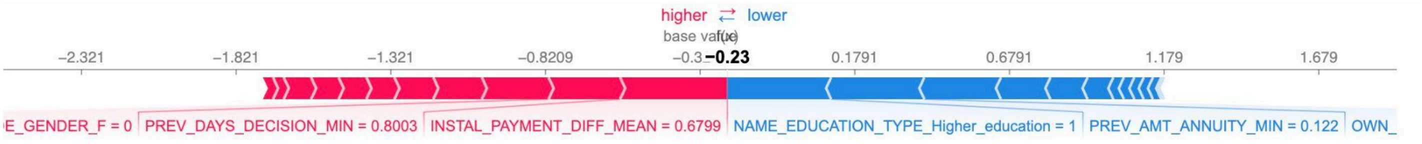
4.1 Global feature importance

The summary plot combines feature importance with feature effects.
Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high. The features are ordered according to their importance.



4.2 Local feature importance

Each feature value is a force that either increases or decreases the prediction. The prediction starts from the baseline. In the plot, each Shapley value is an arrow that pushes to increase (positive value) or decrease (negative value) the prediction. These forces balance each other out at the actual prediction of the data instance. For example, INSTAL_PAYMENT_DIFF_MEAN (Mean difference between the installment value and the amount paid in each installment of the previous credit) contributes to raise the risk of default, whereas NAME_EDUCATION_TYPE_Higher_education being set to 1 lowers the risk.



5. Limits and further improvements

- Study **credit risk literature** and **research papers** to better understand the ratio of costs between Type I Error and Type II Error to implement a more accurate metric.

- **Fine-tune the cost ratio**: as the process of finding the best ratio between two Error Types is very challenging and depends on a variety of factors, it would be interesting to fine-tune it.

- Acquire more knowledge about the banking domain by talking with business stakeholders to create **a better feature engineering** and explore the full potential of the data provided.

- Exclude features that can lead to **discrimination**, like gender, age, region ranking, etc.

- **Automate pipeline creation**: go further in pipeline automation configuration for a more efficient model training.

- **Do not encode categorical features**: LightGBM is able to handle categorical features directly, without the need to one-hot encode them, which can save memory and improve training speed.

- Use **early stopping** to prevent overfitting by monitoring the performance of the model on a validation set.

6. Data Drift analysis

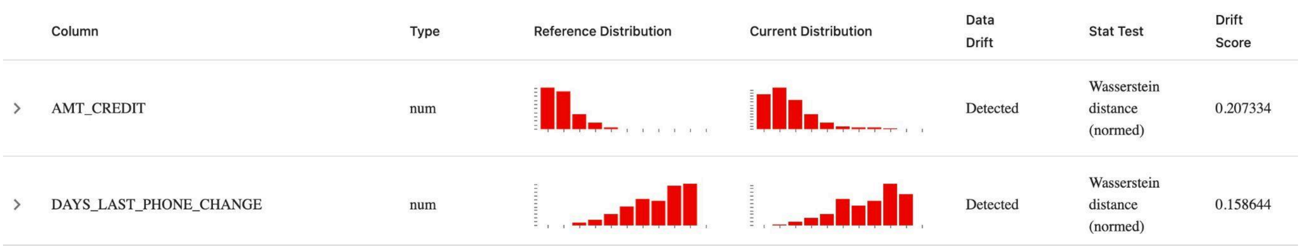
Data Drift refers to the change over time in the statistical properties of the historical data that was used to train a machine learning model. In the real world, data might become outdated causing a different behavior and an accuracy loss of the trained model. That is why it is important to monitor the performance of the model by using a drift detection system and retrain it regularly on updated data to ensure consistent outputs.

The **Data Drift** analysis was performed with the help of the **Evidently** library.

The drift was detected for 8% of columns (2 out of 25). The drift detection threshold being set to 0.5, the detected drift should not impact the model performance.

The list of drifted columns:

1. AMT_CREDIT
2. DAYS_LAST_PHONE_CHANGE



	Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
›	AMT_CREDIT	num			Detected	Wasserstein distance (normed)	0.207334
›	DAYS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.158644

References:

[1] Abdou, H., Pointon, J. Credit scoring and decision-making in Egyptian public sector banks. International Journal of Managerial Finance 5 (4): 391-406., 2009

[2] Bakker Daniel, Odundo, Nyakinda J, Paul Samuel, Performance Evaluation Criteria of Credit Scoring Models for Commercial Lenders. International Journal of Mathematics Trends and Technology (IJMTT) – Volume 65 Issue 7 - July 2019