CREDIT SCORING

Project

Prêt à dépenser

Create a credit scoring calculator based on machine learning techniques

Challenges

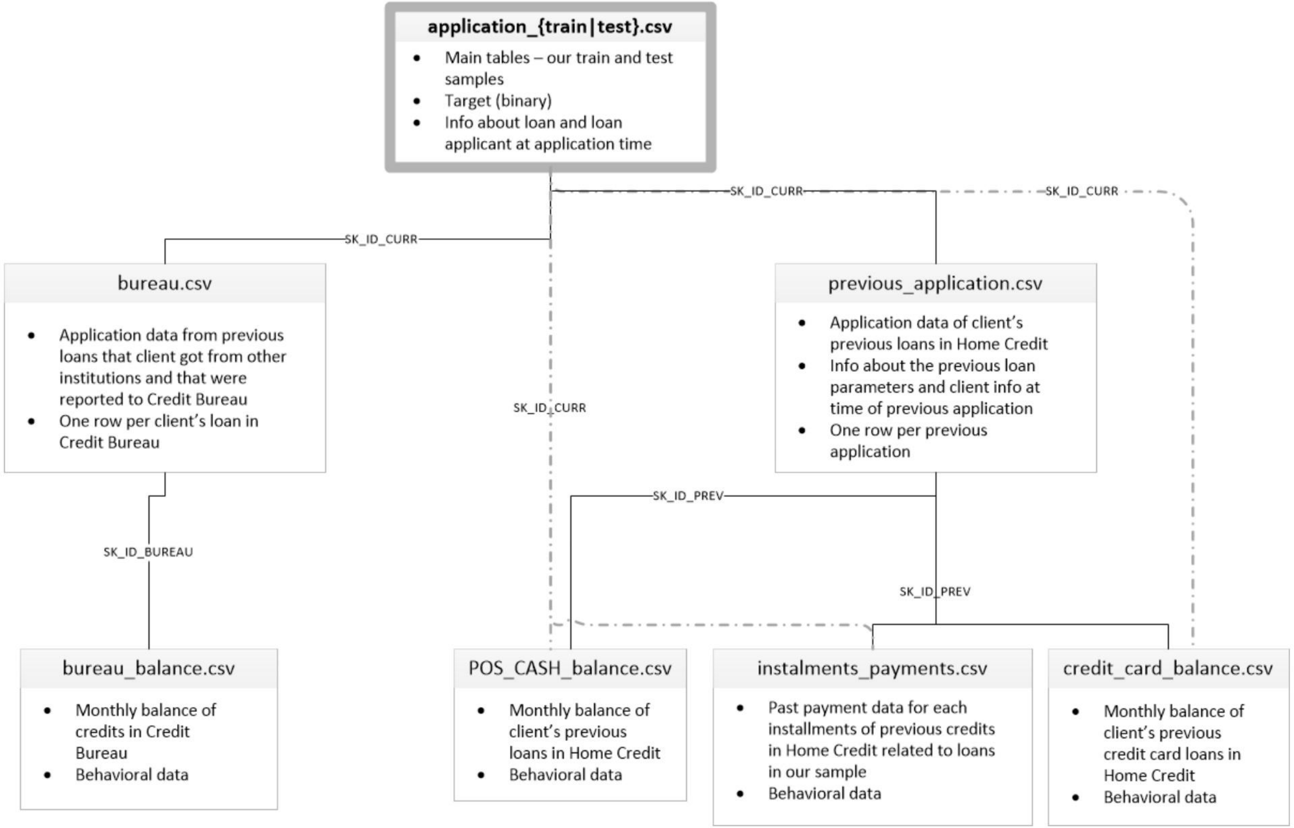
- Clients having poor credit history
- Highly imbalanced data
- Transparency of the credit score

MISSION

- Implement a credit scoring model
- Create a predict API
- Create an interactive dashboard

DATA

Database schema



Quick overview

- **Data Origin:** Home Credit Kaggle Competition
- 7 tables
- 307511 applications in the train set
- 8% applications with payment difficulties
- 92% of no risk applications

METHODOLOGY

2 TYPES OF ERRORS

TYPE I

FALSE POSITIVES

- customers predicted as potential defaulters but did not default
- affect profitability

1:10

TYPE II

FALSE NEGATIVES

- customers predicted as non risky and who defaulted
- important money loss

EVALUATION METRICS

1	COST RATIO SCORE	Ratio of 10*FN + FP
2	Fβ-SCORE (β = 10)	Give more weight to Recall
3	ROC AUC	Trade-off between True Positive Rate and the False Positive Rate
4	RECALL	True Positive Rate
5	ACCURACY	Proportion of observations that were correctly predicted

MODELS

DUMMY: baseline

- Dummy Classifier

LINEAR

- Logistic Regression

ENSEMBLE

- Random Forest Classifier

GRADIENT BOOSTING

- LightGBM
- XGBoost

TRAINING PROCESS FOR EACH MODEL

STEP 1

CLASS IMBALANCE

Choose the best method for the class balancing

STEP 2

FEATURE ENGINEERING

Feature transformations + optimization

STEP 3

PREPROCESSING

Selecting the right preprocessing techniques

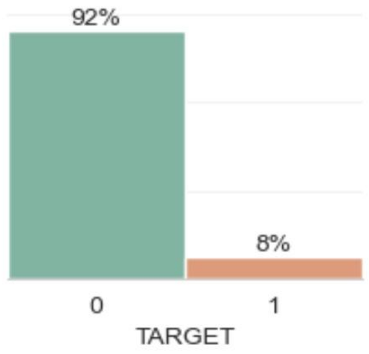
STEP 4

FINE-TUNING

Choosing optimal hyperparameters

CLASS IMBALANCE

- Oversampling
  - SMOTE
  - SMOTE-NC
- Class weights
  - `class_weights`: `balanced’ (Logistic Regression, Random Forest Classifier, LightGBM)
  - `scale_pos_weight`: 11 (XGBoost)



FEATURE ENGINEERING

STEP 1

Merge datasets progressively

STEP 2

Perform aggregations + functions

STEP 3

Optimize dataset memory usage (by 47%)

STEP 4

Select most important features (789 → 25)

FEATURE OPTIMIZATION

- Drop features:
  - with low variance (0.1 variance threshold)
  - correlated (>= 0.7)
- Select 25 final features:
  - Logistic Regression *coef_*
  - LightGBM *feature_importances_*
  - XGBoost *feature_importances_*

PREPROCESSING

OUTLIERS	MISSING VALUES	ENCODING	SCALING	NORMALIZATION
IQR factor 1.5 IQR factor 2.5	Drop columns with > 60% of NaN Simple Imputer: ▪ median ▪ most frequent	One Hot Encoding (categorical)	StandardScaler MinMaxScaler RobustScaler	Power Transformer

FINE TUNING

01	GridSearchCV	all parameter combinations
02	RandomizedSearchCV	randomized search over parameters
03	HyperOpt	Bayesian Optimisation

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**

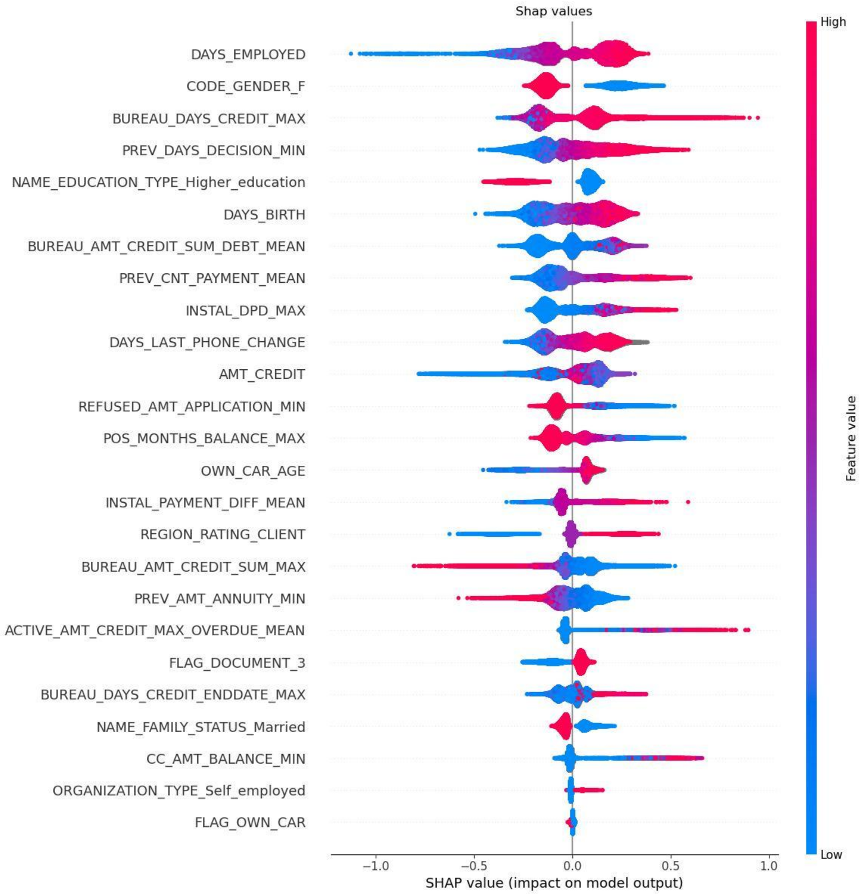
LIGHTGBM CONFIGURATION

Class balance setting: class_weights: ‘balanced’

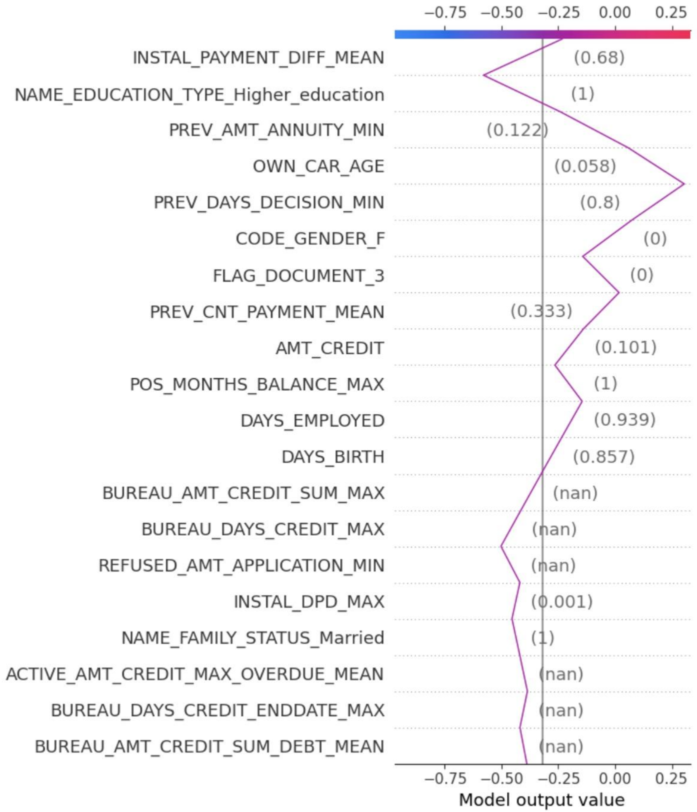
Best threshold: 0.5

PREPROCESSING	HYPERPARAMETERS
▪ Filling missing values: × ▪ Removing outliers: × ▪ OneHotEncoding: ✓ ▪ Scaling (MinMaxScaler): ✓ ▪ Normalization: ×	▪ n_estimators: 158 ▪ num_leaves: 111 ▪ max_depth: 4 ▪ learning_rate: 0.09 ▪ min_data_in_leaf: 135 ▪ subsample_for_bin: 20000 ▪ colsample_by_tree: 0.59 ▪ lambda_l1: 0.71 ▪ lambda_l2: 0.19

FEATURE IMPORTANCE: GLOBAL



FEATURE IMPORTANCE: LOCAL

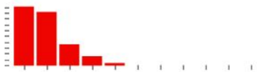
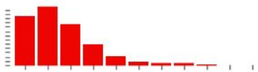
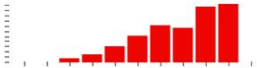
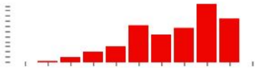


DATA DRIFT ANALYSIS

Tool: Evidently

Detected drift: 8.0% of columns (2 out of 25)

Detection threshold: 0.5

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

CONTINUOUS INTEGRATION

MODEL TRACKING

- MLFLow

VERSION CONTROL

- Git
- Github (repository link)

JOBS

- Tests with PyTest
- Github actions

API HOSTING

- Render platform

DASHBOARD

- Streamlit (dashboard link)

COMMITS

feat(ops): generate a data drift report
NatChe committed last week · ✓ 1 / 1

refacto(dashboard): move dashboard.py to the root
NatChe committed last week · ✓ 1 / 1

fix(dashboard): run shap initjs
NatChe committed last week · ✓ 1 / 1

fix(dashboard): use abs path for feature importance image
NatChe committed last week · ✓ 1 / 1

fix(dashboard): use absolute path for css
NatChe committed last week · ✓ 1 / 1

GITHUB ACTIONS

feat(ops): generate a data drift report
PyTest #21: Commit 10e73c1 pushed by NatChe

refacto(dashboard): move dashboard.py to the root
PyTest #20: Commit a64cedd pushed by NatChe

fix(dashboard): run shap initjs
PyTest #19: Commit 116775f pushed by NatChe

fix(dashboard): use abs path for feature importance image
PyTest #18: Commit e8686ab pushed by NatChe

fix(dashboard): use absolute path for css
PyTest #17: Commit 209fc45 pushed by NatChe

TO GO FURTHER

- Study credit risk literature and research papers (feature engineering, score).
- Fine-tune Cost Ratio between Error Type I and Error Type II.
- Exclude features that can lead to discrimination.
- More advanced optimization of the LightGBM model.

THANKS!

Links:

1. Github repository: https://github.com/NatChe/credit-scoring-dashboard
2. Dashbord on Streamlit:
   https://credit-scoring-dashboard-vo4t4kjrntvwp3fabtzsss.streamlit.app/
3. Render platform: https://render.com/
4. Kaggle notebook used for Feature Engineering:
   https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features
5. Main page of the Home Credit Kaggle Competition for the data explanation:
   https://www.kaggle.com/competitions/home-credit-default-risk/data