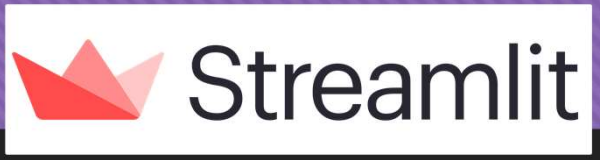
P7 Datascientist – OpenClassrooms
Etienne Lardeur
Mentor : Xavier Tizon
Evaluateur : Mohammed Sedki

Cash & Revolving loans for clients with *no* or *few* loans history

Develop ***scoring model*** of the applicant's ***risk of failure to repay loan***
with ***prediction interpretation***, through an ***interactive dashboard***
to argue whether loan is **granted** or **rejected** and help studying **why**.

Project's deliverables



+ 5 Jupyter Notebooks:
- **P7_EDA**: Exploratory Data Analysis,
- **P7_FE**: Feature engineering
- **P7_FS**: Feature Selection
- **P7_Model** : Scoring & Model Evaluation,
- **P7_Interpretation** : Model Interpretation

+ Methodological notice (FR)

https://github.com/EtienneLardeur/P7_Scoring

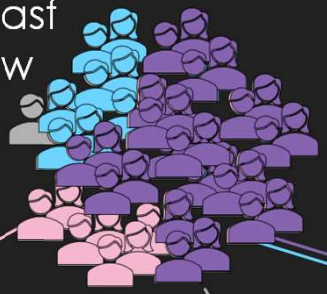
+ **Dashboard** : local_app.py or remote
https://share.streamlit.io/etiennelardeur/streamlit_app/main/local_app.py

https://github.com/EtienneLardeur/Streamlit_App

Table of contents

Use case

Applicants
307 511 past
48 744 new



(New)
Client's ***profile***
& *payment difficulties* 0/1 (Past)

Client's "***internal***" applications incl. **rejected** (95.03 %)

Client's "***external***" loans (85.84 %)

Client's ***both*** loans' knowledge (81.63 %).

Anteriority of 8 years ***sliding*** *from current application*

Application

Growth intentions,
Liquidity & key rates,
Economic health, *etc.*

Global Business Context

Application Analysis

Scoring Model & Interpretability

2 Dashboard

1 Predict "**0**" Good | Predict "**1**" Bad

Machine Learning

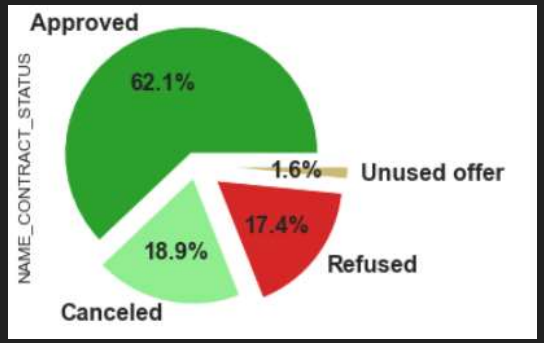
a. Inputs
b. Models
c. Results

Learn *from the past & **Predict** failure*

Decision

Keep reason

Detect failure… & learn from it !



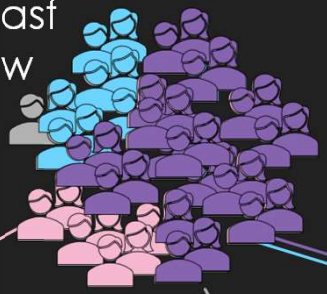
Use case

1. Use case & inputs
2. Scoring & Integration
3. Model
4. Dashboard

Growth intentions,
Liquidity & key rates,
Economic health, *etc.*

Global Business Context

Applicants
307 511 past
48 744 new



Application

Application Analysis

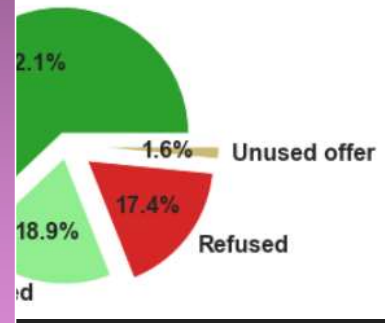
Decision

Keep reason

Detect failure… & learn from it !

Business Goals should drive our study, defining how we intend to use the model and its interpretability:

- Assess **predictions** vs **reality**,
- Challenge **usual rules** with **model interpretations**,
- Define and test **risk mitigation** actions.



we have no time-stamp nor contextual data to catch repayment failure as "events"… rather than "fix" we would "prevent" defaults by focusing our monitoring

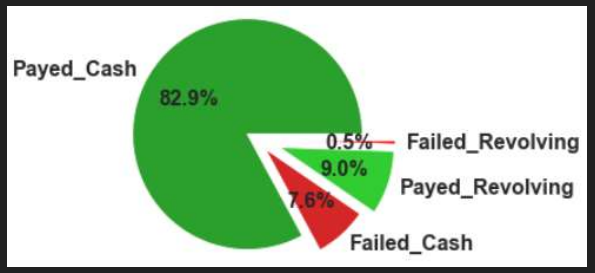
Inputs: overview

Target meaning:

1 - *client with payment difficulties: he/she had late payment more than **X** days on at least one of the first **Y** installments of the loan in our sample,*
0 - *all other cases*

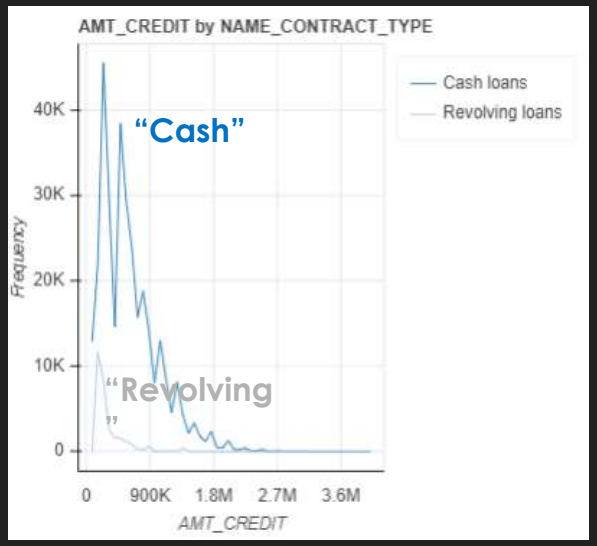
Failure rates:

- ***Global : 8.07%***
- ***Revolving 5.48%***
- ***Cash 8.35%***

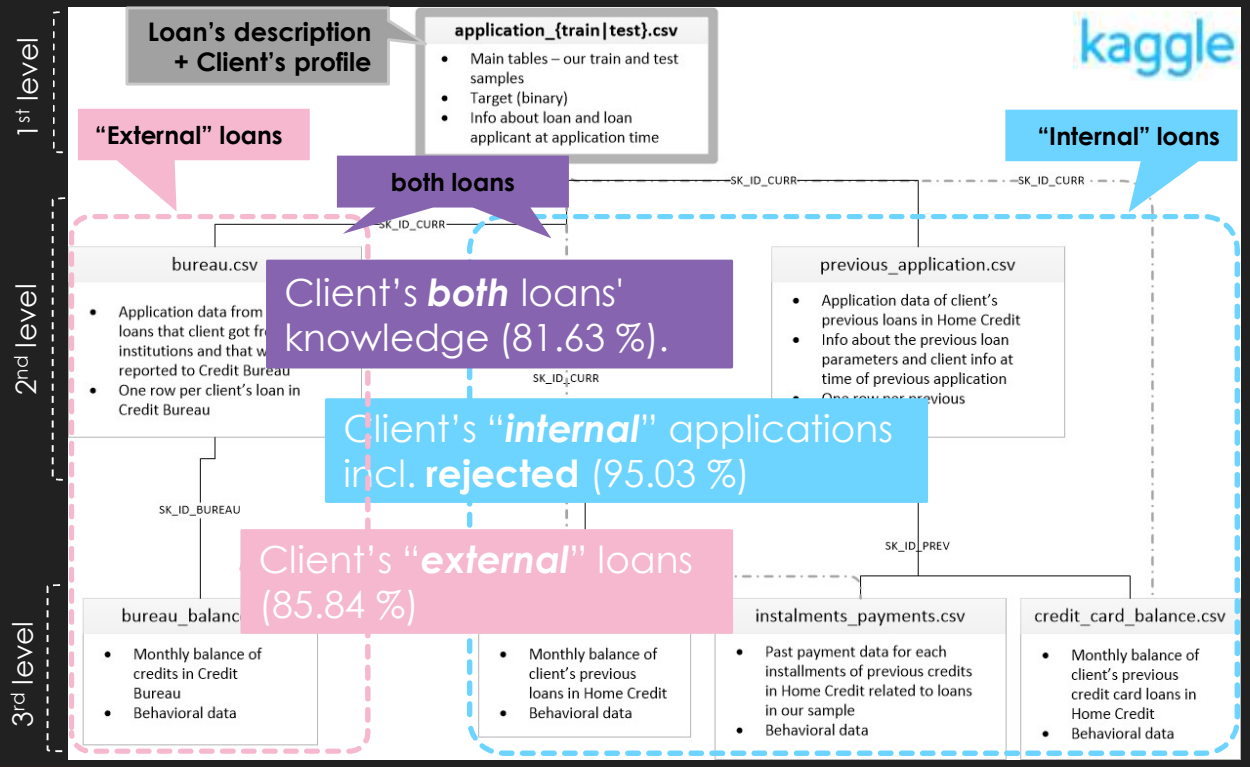


Credit Amounts:

"**Revolving**" loans implies **lower amounts** and **ability** to cut down further non repaid expenses than for "**Cash**" loans.



Profile & Anteriority knowledge: 7 data sources



Kaggle Kernel re-use: limitations

While Competitors focus on max score "by any means", we must provide **interpretable** decisions, including the possibility given to customers to **explore** their case.

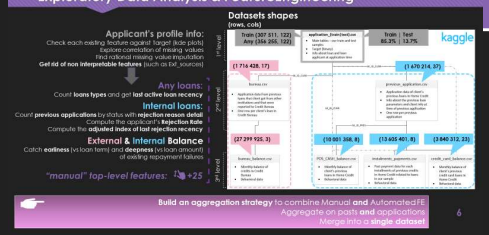
Inputs: analysis & enhancement

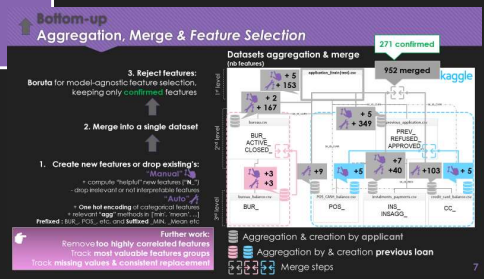
interdependence

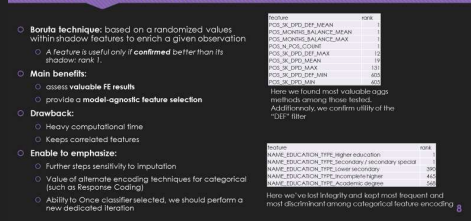
1. Top-Down
EDA & Manual FE

2. Bottom-Up
Manual & Auto FE
& Aggregation

3. Model-agnostic
Feature Selection







Further work:
Iterate according to **best classifier's** features importance
Take into account **interpretability feedbacks**

Model + Interpretation

Top-down
Exploratory Data Analysis & *FeatureEngineering*

Applicant's profile info:

Check each existing feature against Target (kde plots)
Explore correlation of missing values
Find rational missing value imputation
Get rid of non interpretable features (such as Ext_sources)

Any loans:

Count **loans types** and get **last active loan recency**

Internal loans:

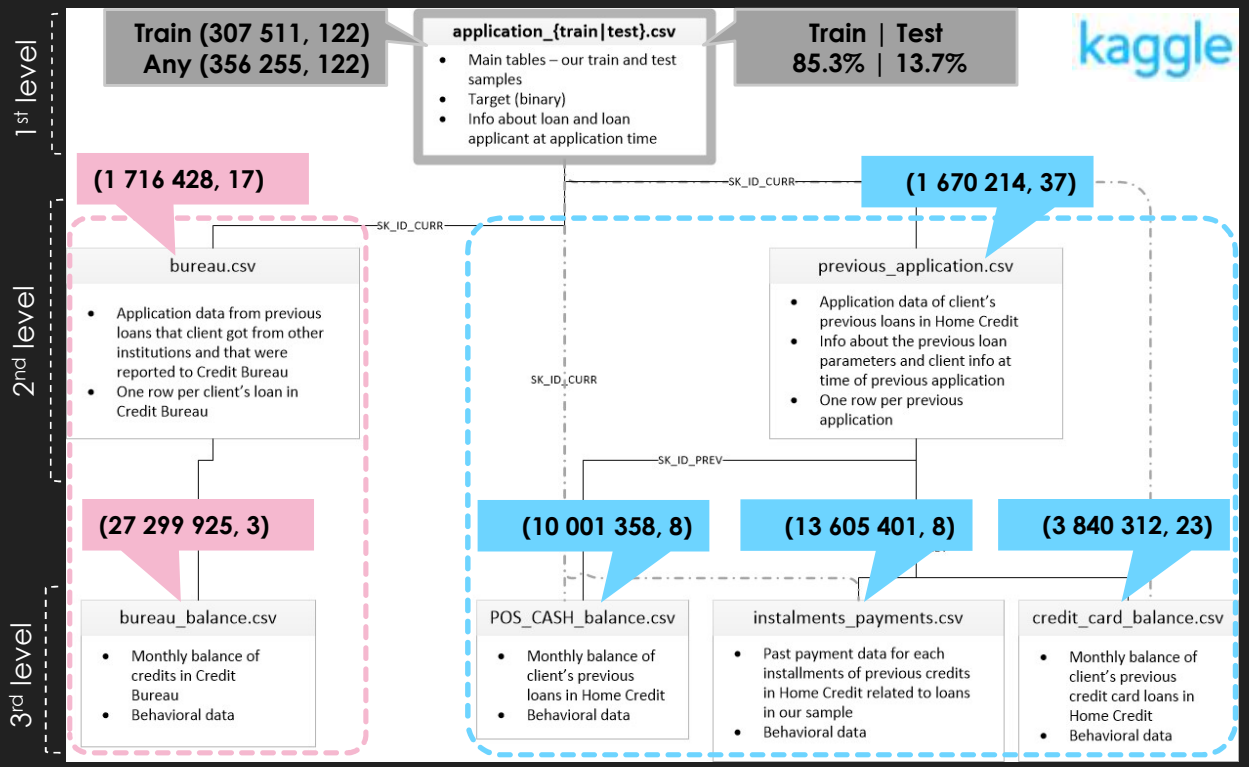
Count **previous applications** by status with **rejection reason detail**
Compute the applicant's **Rejection Rate**
Compute the **adjusted index of last rejection recency**

External & Internal Balance

Catch **earliness** (vs loan term) and **deepness** (vs loan amount) of existing repayment failures

"manual" top-level features: +25

Datasets shapes
(rows, cols)



Build an aggregation strategy to combine Manual **and** Automated FE
Aggregate on pasts **and** applications
Merge into a **single dataset**

Bottom-up
Aggregation, Merge & *Feature Selection*

3. Reject features:
Boruta for model-agnostic feature selection, keeping only **confirmed** features

2. Merge into a single dataset

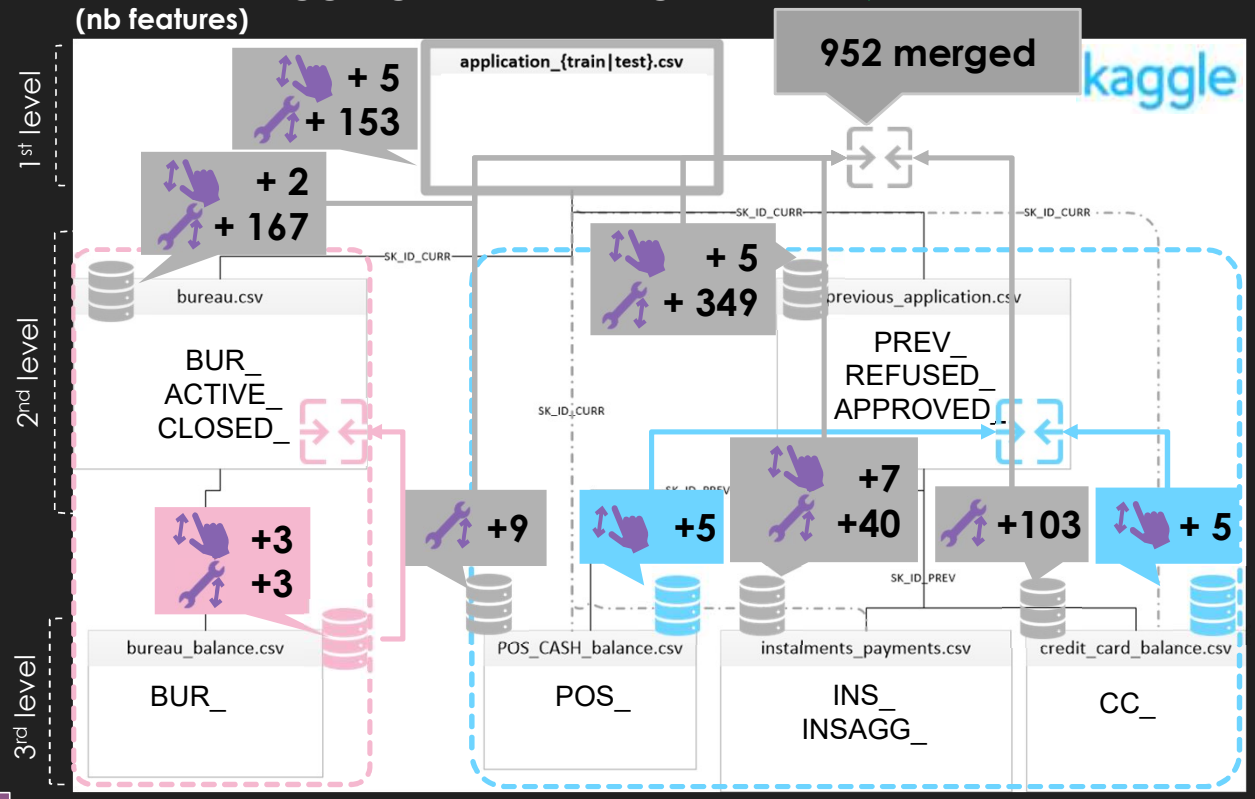
1. Create new features or drop existing's:

"Manual"
+ compute "helpful" new features ("**N**_")
- drop irrelevant or not interpretable features

"Auto"
+ **One hot encoding** of categorical features
+ relevant "**agg**" methods in ['min', 'mean', ...]
Prefixed : BUR_, POS_, etc, and **Suffixed** _MIN, _Mean etc

Further work:
Remove **too highly correlated features**
Track **most valuable features groups**
Track **missing values & consistent replacement**

Datasets aggregation & merge
(nb features)



Aggregation & creation by **applicant**

Aggregation by & creation **previous loan**

Merge steps

Track most valuable features

- **Boruta technique:** based on a randomized values within shadow features to enrich a given observation
  - *A feature is useful only if **confirmed** better than its shadow: rank 1.*
- **Main benefits:**
  - assess **valuable FE results**
  - provide a **model-agnostic feature selection**
- **Drawback:**
  - Heavy computational time
  - Keeps correlated features
- **Enable to emphasize:**
  - Further steps sensitivity to imputation
  - Value of alternate encoding techniques for categorical (such as Response Coding)
  - Ability to Once classifier selected, we should perform a new dedicated iteration

feature	rank
POS_SK_DPD_DEF_MEAN	1
POS_MONTHS_BALANCE_MEAN	1
POS_MONTHS_BALANCE_MAX	1
POS_N_POS_COUNT	1
POS_SK_DPD_DEF_MAX	12
POS_SK_DPD_MEAN	19
POS_SK_DPD_MAX	131
POS_SK_DPD_DEF_MIN	605
POS_SK_DPD_MIN	605

Here we found most valuable aggs methods among those tested. Additionnaly, we confirm utility of the "DEF" filter

feature	rank
NAME_EDUCATION_TYPE_Higher education	1
NAME_EDUCATION_TYPE_Secondary / secondary special	1
NAME_EDUCATION_TYPE_Lower secondary	390
NAME_EDUCATION_TYPE_Incomplete higher	465
NAME_EDUCATION_TYPE_Academic degree	568

Here we've lost integrity and kept most frequent and most discriminant among categorical feature encoding

Table of contents

1 • Use case & inputs
2 • Scoring & Integration
3 • Model
4 • Dashboard

Scoring & integration

Confusion Matrix	Actual 0 Good	Actual 1 Bad
Predict 0 Good	tn Usual business	fn High risk exposure
Predict 1 Bad	fp Loose clients	tp Usual business

Y

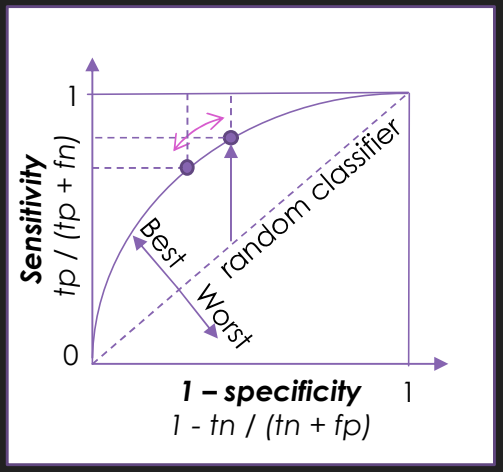
N

- The Confusion Matrix -

- According to the target definition 1 - client failed to repay loan, 0 – no difficulty:
  - Trusting a **negative prediction** in case of **tn: true negative** is usual business, but a **fn: false negative** means exposure to a higher risk of defaults.
  - Trusting a **positive prediction** in case of **tp: true positive** is usual business, but a **fp: false positive** means losing a "good" client.

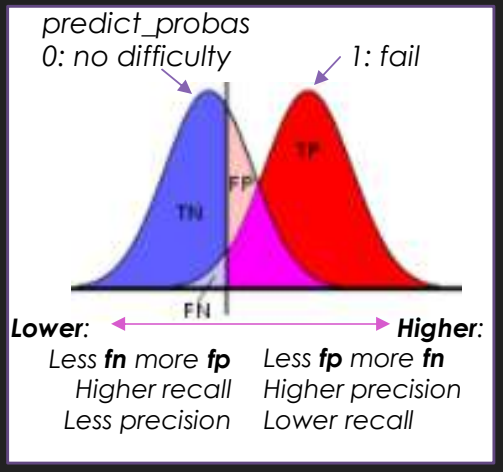
→ **We want to minimize (fn + fp), with higher interest on fn**

1. a classifier is basically as good as its Area under ROC Curve is large



- The ROC Curve -

2. Given a certain classifier, we can tune threshold initially set to 0.5, in order to optimize confusion matrix



- Threshold adjustment -

3. Minimizing **fn** will consequently increase **fp** and implies to monitor another score, such as an Fbeta

$$F_\beta = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$$

With : . Precision = tp / (tp + fp)

. Recall = tp / (tp + fn)

$$F_\beta = \frac{(1+\beta^2)\cdot\text{true positive}}{(1+\beta^2)\cdot\text{true positive}+\beta^2\cdot\text{false negative}+\text{false positive}}$$

Best value of threshold will depend on value of **beta**, which approximately measure the n times more important are the damage

- Alternate scoring, to find the "best threshold" -

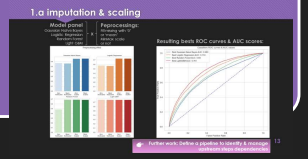
Table of contents

1 • Use case & inputs
2 • Scoring & Integration
3 • Model
4 • Dashboard

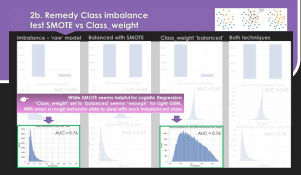
Find the best model: process

1. Assess preprocessing relevance

a. imputation, scaling



b. Class imbalance remediation



2. Classifiers evaluation & hyperparam tuning

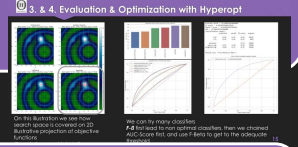
HYPEROPT

a. Broaden search with any classifier

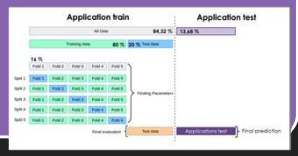
b. "chain" objective functions

1. AUC → **2. *F-ß***

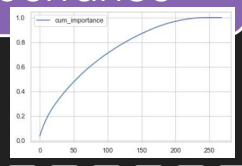
3. Focus on LightGMB hyperparameters optimization



Cross-validation

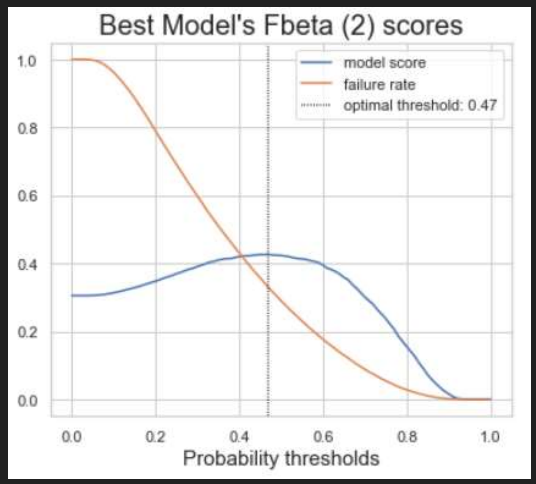


Simplification by cumulative feature importance



4. Train best & determine "best" threshold via domain optimization

Best : LGBMClassifier
AUC score: 0.7646



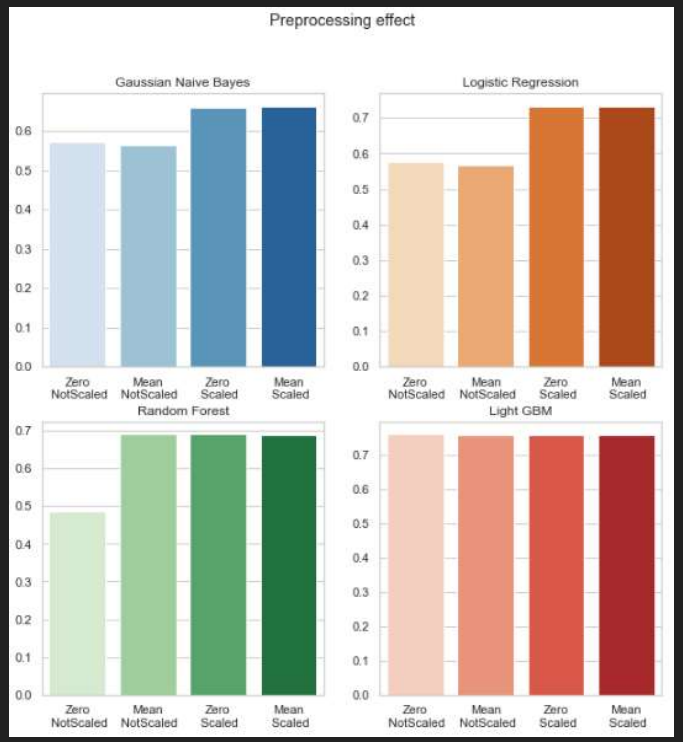
- Threshold trade-off – Failure rate versus best Fbeta

1.a imputation & scaling

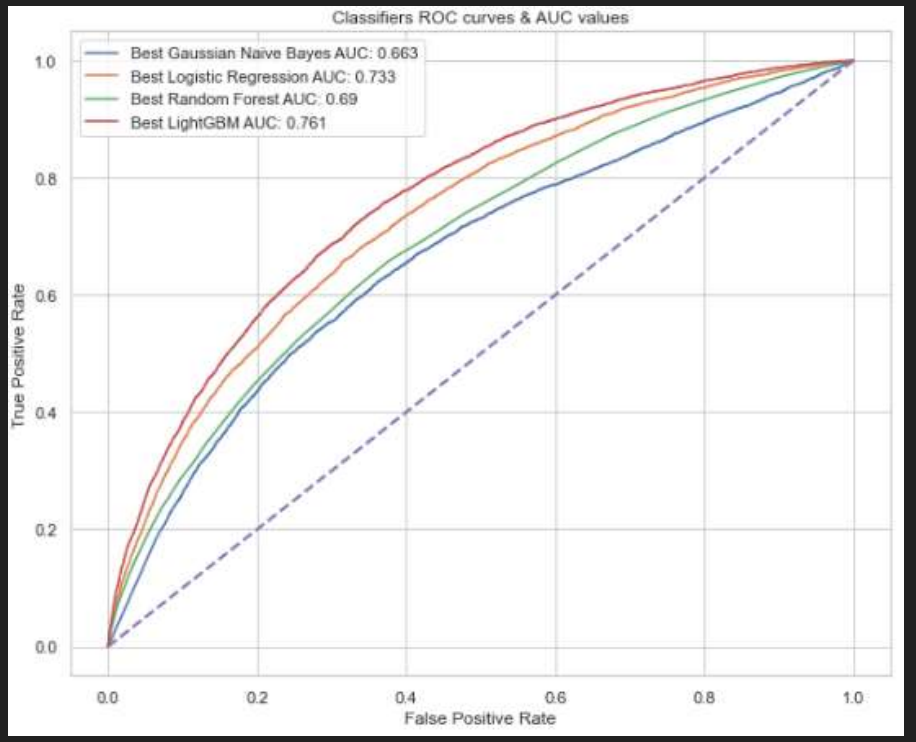
Model panel
Gaussian Naïve Bayes
Logistic Regression
Random Forest
Light GBM

X

Peprocessings:
Fill missing with '0'
or 'mean'
MinMax scale
or not

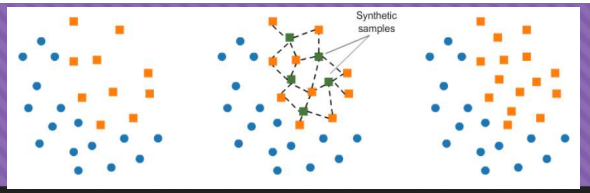


Resulting bests ROC curves & AUC scores:

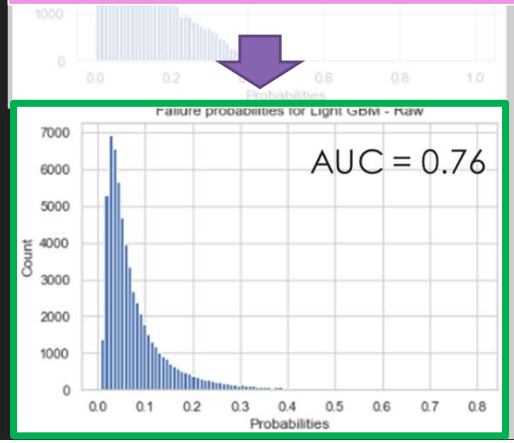


Further work: Define a pipeline to identify & manage upstream steps dependencies

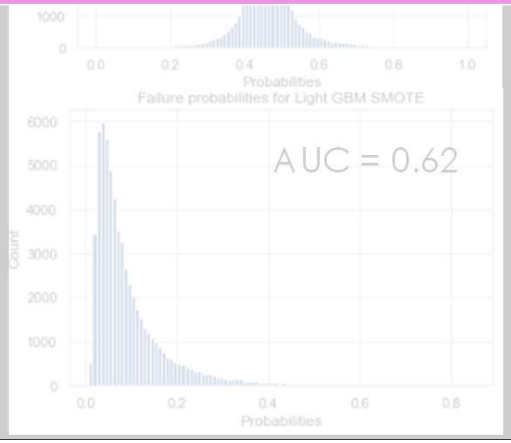
2b. Remedy Class imbalance test SMOTE vs Class_weight



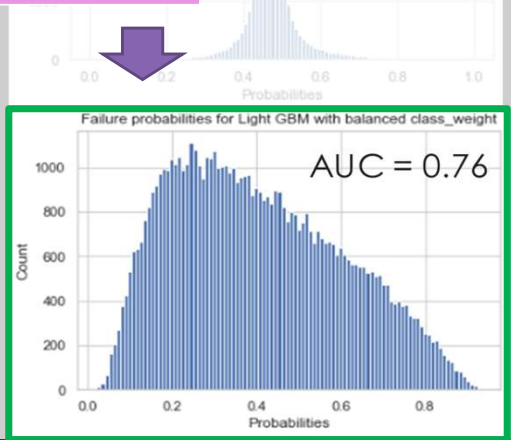
Imbalance – 'raw' model



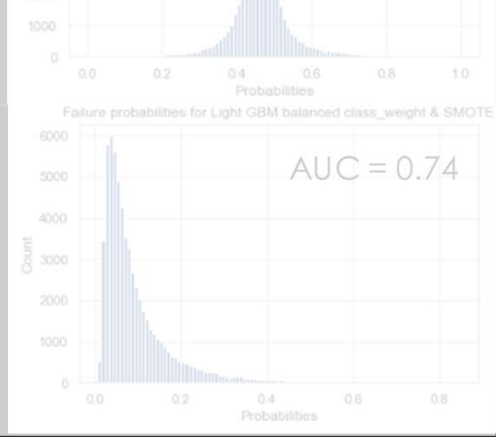
Balanced with SMOTE



Class_weight 'balanced'

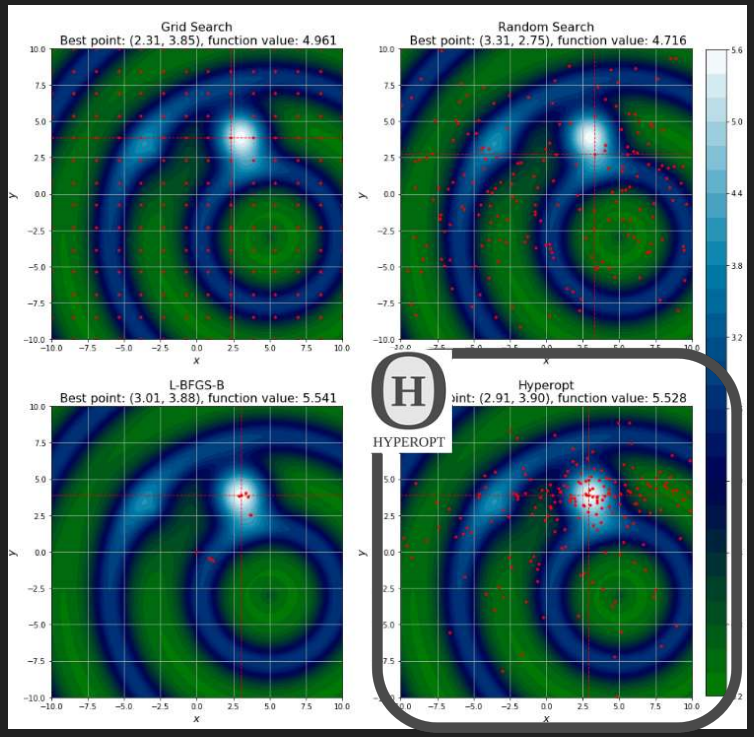


Both techniques

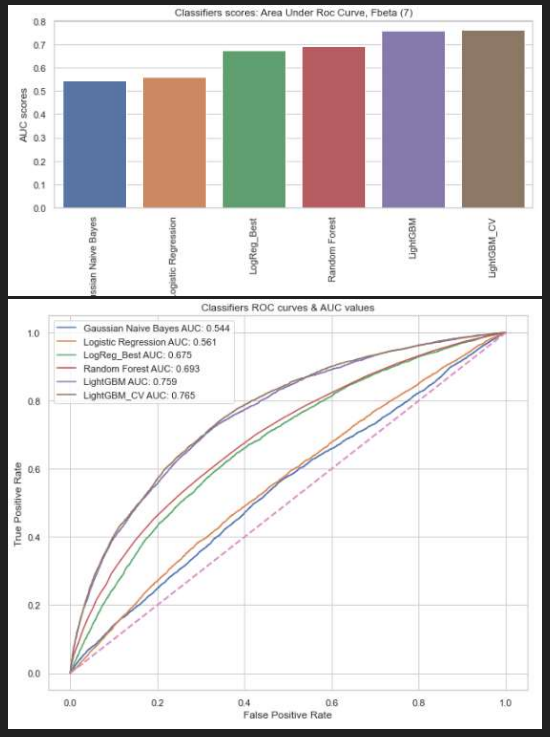


While SMOTE seems helpful for Logistic Regression
"Class_weight" set to 'balanced' seems "enough" for Light GBM,
With even a rough behavior able to deal with such imbalanced class

3. & 4. Evaluation & Optimization Hyperopt with CV



On this illustration we see how search space is covered on 2D illustrative projection of objective functions
(Kaggle hyperopt tutorial)



We can try many classifiers ***F-ß*** first lead to non optimal classifiers, then we chained AUC-Score first, and use ***F-ß*** to find best threshold

Best model:

LGBMClassifier
(boosting_type='goss',
class_weight='balanced',
learning_rate=0.019…,
max_depth=23,
n_estimators=610,
num_leaves=52)
Best AUC score: 0.7646
With Fbeta score (7): 0.6118

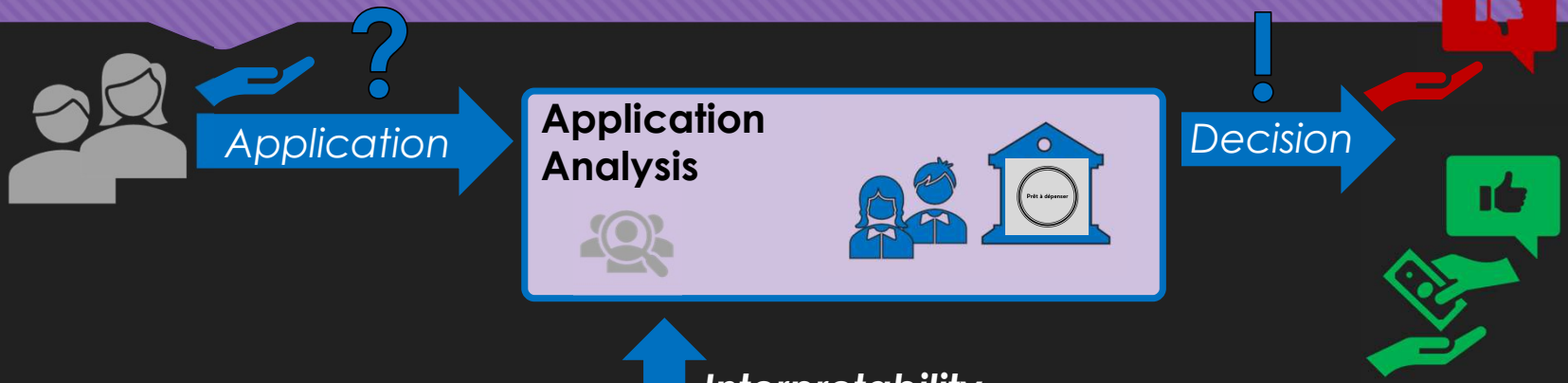
+ Confirmed on 80% model cumulative importance (125 features)
Best AUC score: 0.764
With Fbeta score (7): 0.6105

Imputed by mean, unscaled

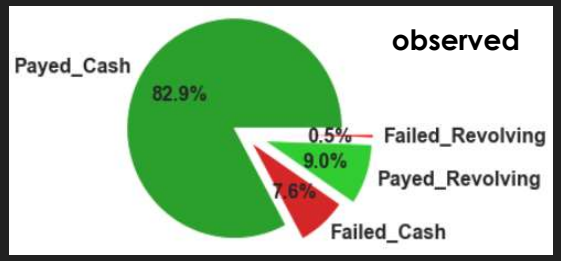
Table of contents

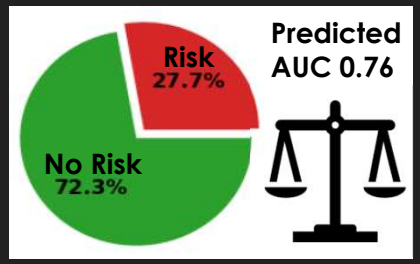
1 • Use case & inputs
2 • Scoring & Integration
3 • Model
4 • Dashboard

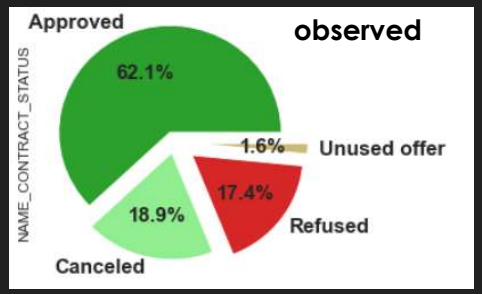
Go back to the Use Case



Done! now open the **Black Box**

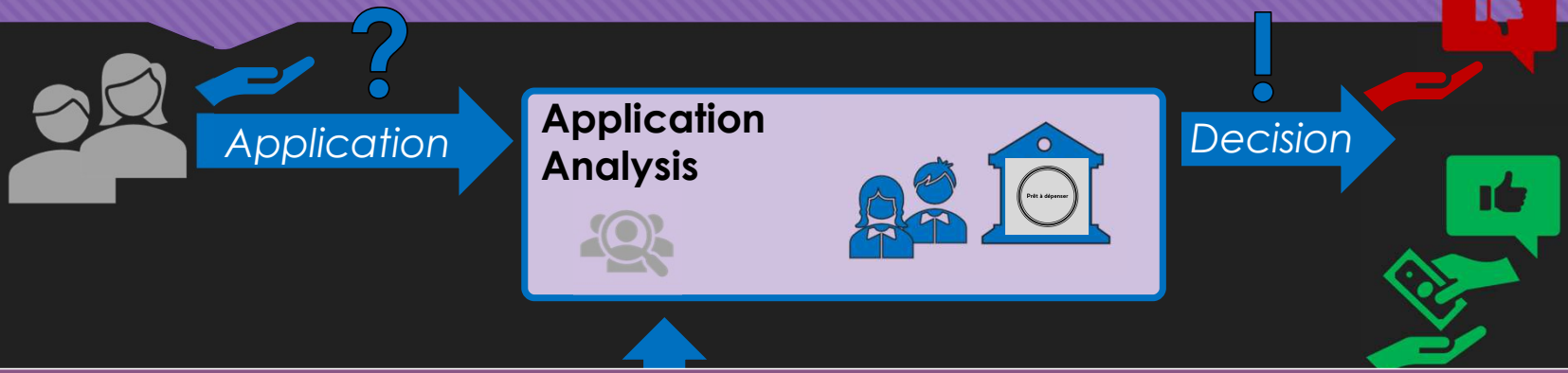






Thresholding as a trade-off

Go back to the Use Case



study of model **interpretability**, to enable business goals such as :

- Assess **predictions** vs **reality**,
- Challenge **usual rules** with **model interpretations**,
- Define and test **risk mitigation** actions.

Providing a **Dashboard**

Dashboard

Build Dashboard

1. • Use case & inputs
2. • Scoring & Integration
3. • Model
4. • Dashboard

Display samples from both Class (demo only)

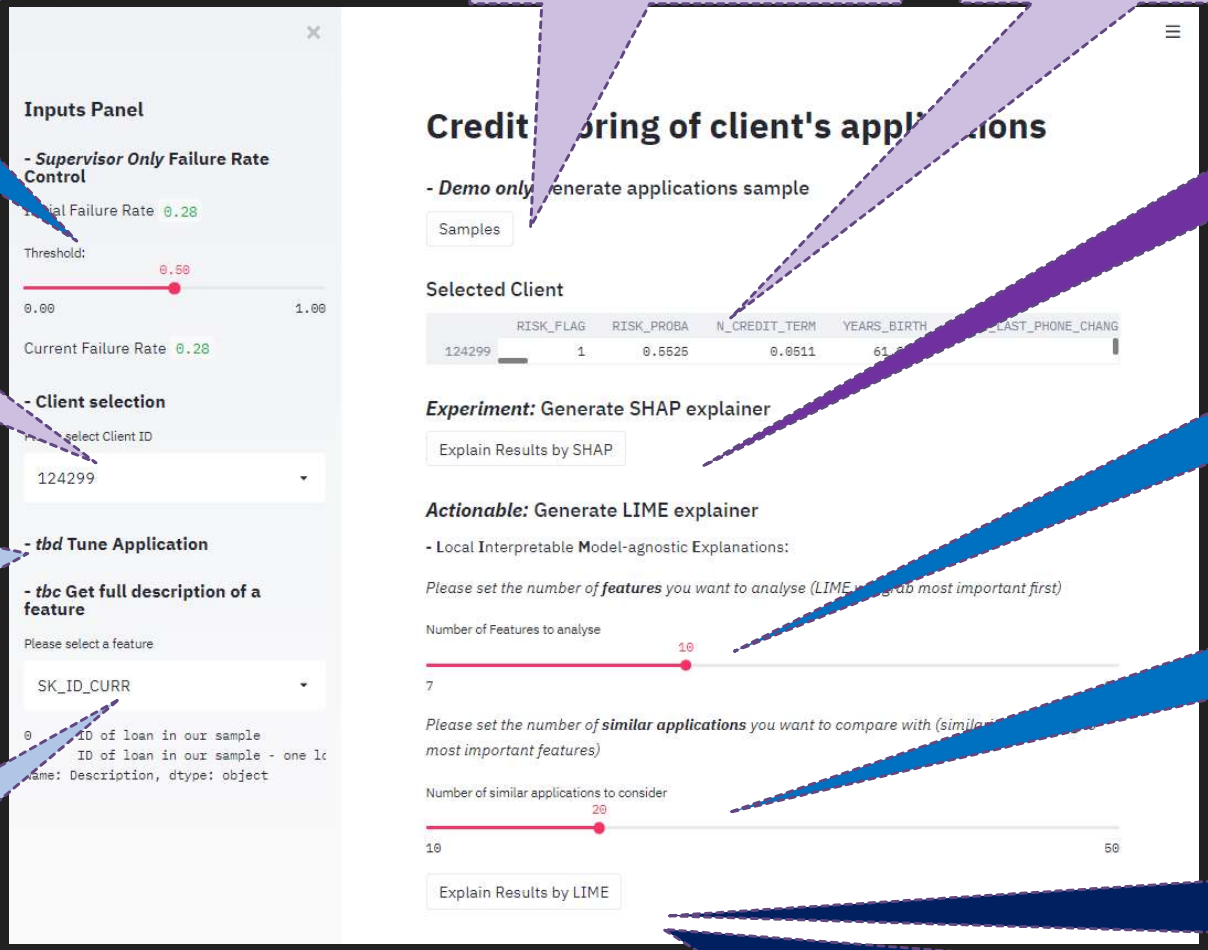
Display selected application details

Adjust threshold and check failure rate

Filter client's application

(tbd) Tune application & actualize prediction

(tbc) Provide features plain text description



Display **SHAP** interpretation *passive results*

Select the number of features to analyse (sorted by importance)

Select the size of neighborood (number of similar applications)

Display **LIME** interpretation *interactive results*

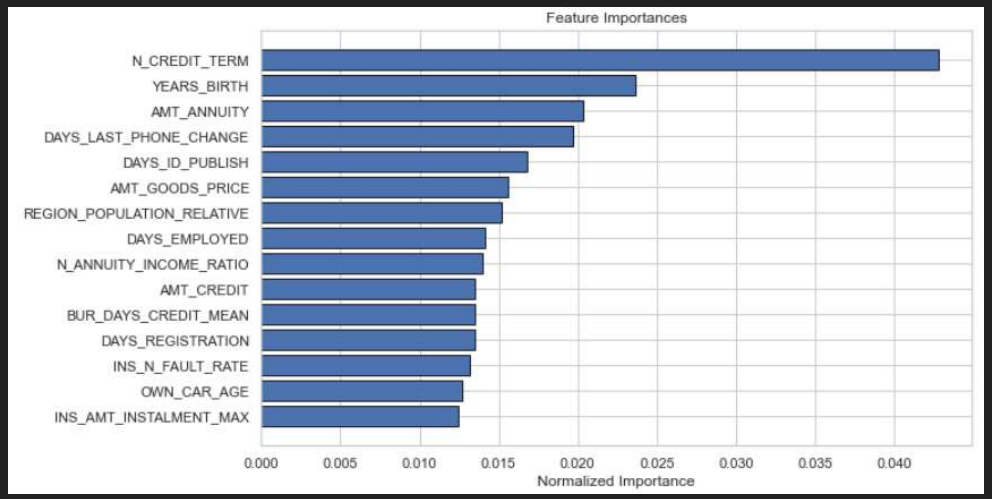
Compare client's detail with mean of Class 0, Class 1 and similar applications

Dashboard: interpretability with "explainers"

Model Interpretability Does Not Mean Causality

1. A first step in the field of **interpretability** comes with the model's **features importance**. This global insight is ***not enough*** for an individual application study.

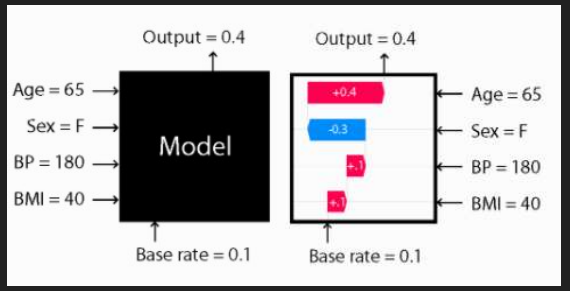
LightGBM most important features



Gini Importance, *or* ***Mean Decrease in Impurity****, summed over the number of splits (accross all tress) that include the feature, proportionaly to the number of samples it splits.*

2. To enable local interpretability let's introduce LIME and SHAP techniques

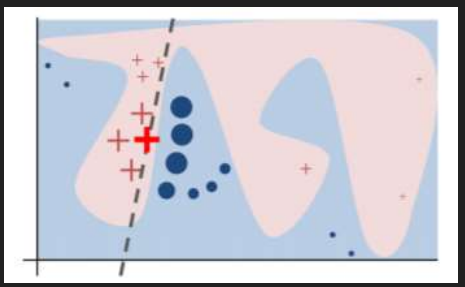
SHAP



- **Explainer f(model)**
- **average of the marginal contributions across all permutations** (how heavy are the contribution of a feature to the loss).
- Enable *global and local interpretability*
- compatible with tree-based model

https://shap.readthedocs.io/

LIME



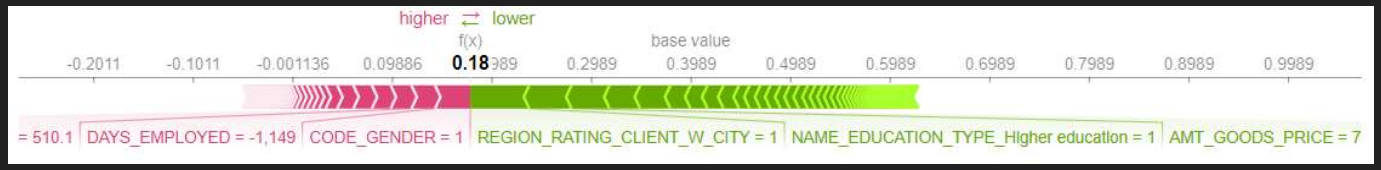
- **Explainer != f(model)**
- **samples with distance-weight to provide a local interpretation**
- Dedicted packages for text, image, etc.

https://lime-ml.readthedocs.io/
https://arxiv.org/pdf/1602.04938.pdf

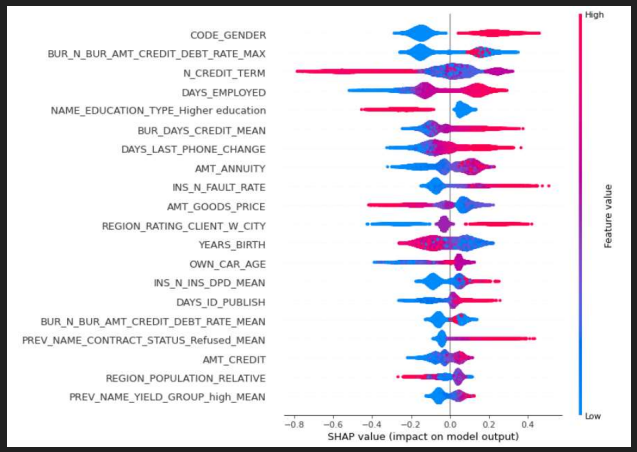
SHapley Additive exPlanations

Force plot of class 1 - Individual

Force plot of class 1 - Collective



Summary plot of class 1 - Collective

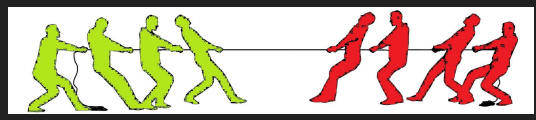


Summarize the fitted model behavior,
either considering binary class or predict probability

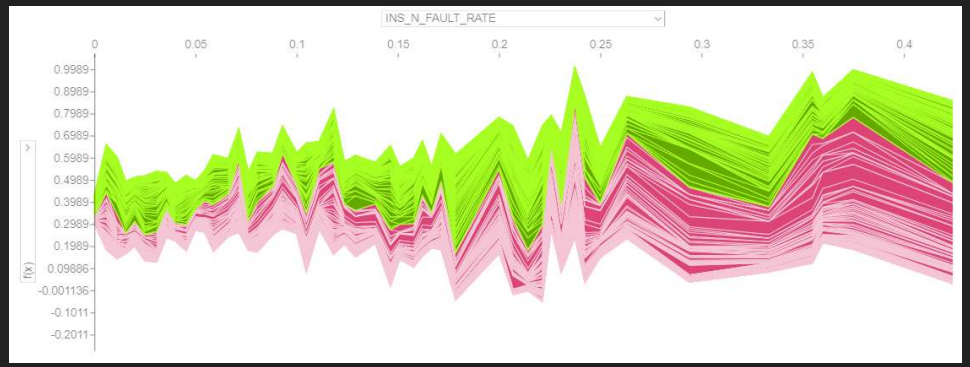
Interpretation

Shows how features impacts prediction

- Considering a single prediction
- Or any prediction



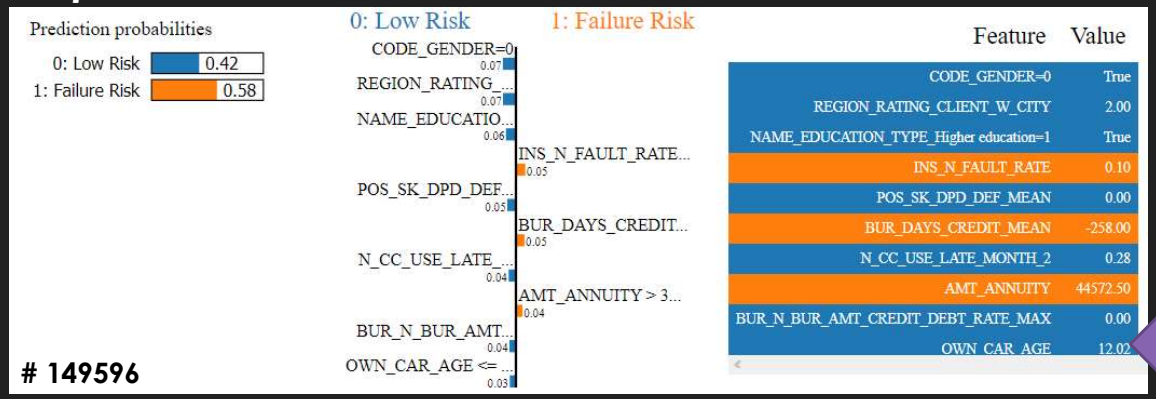
Force plot – collective (sample 500)
Case of the previous instalments fault rate



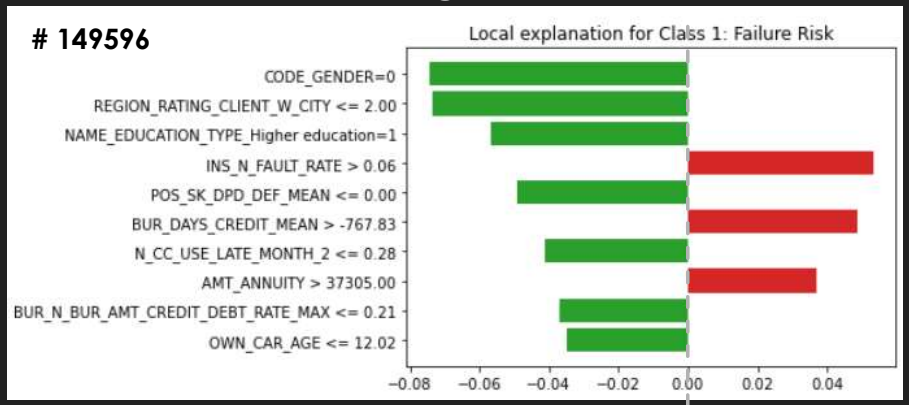
Enable interactive feature analysis
Here we catch the trend of feature impact

Local Interpretable Model-Agnostic Explanations

Explanation as shown in notebook - Individual



Explanation as plot figure – Class 1

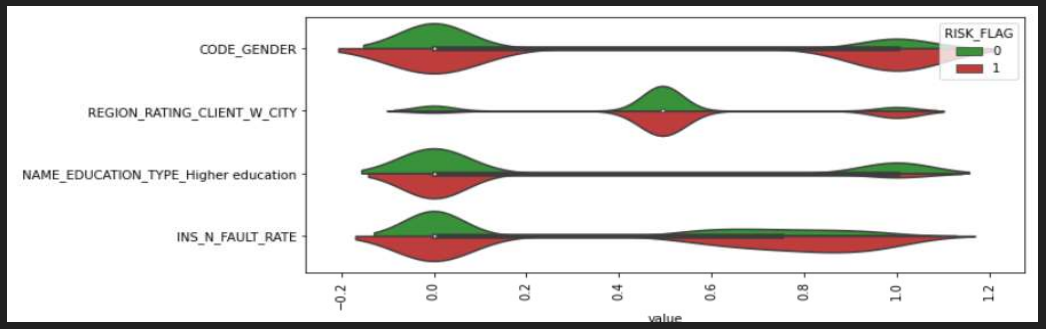


Usually Green for Class support, but in our case Class 1 is Risk of failure

contradicts | **supports**

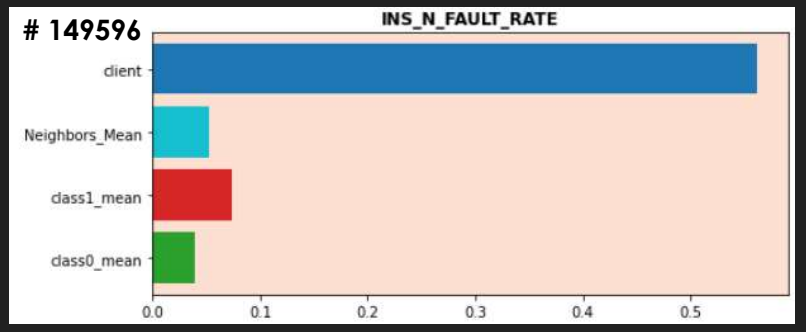
Interpretation

Overal values distribution, by class



While having "good" values for the first three features

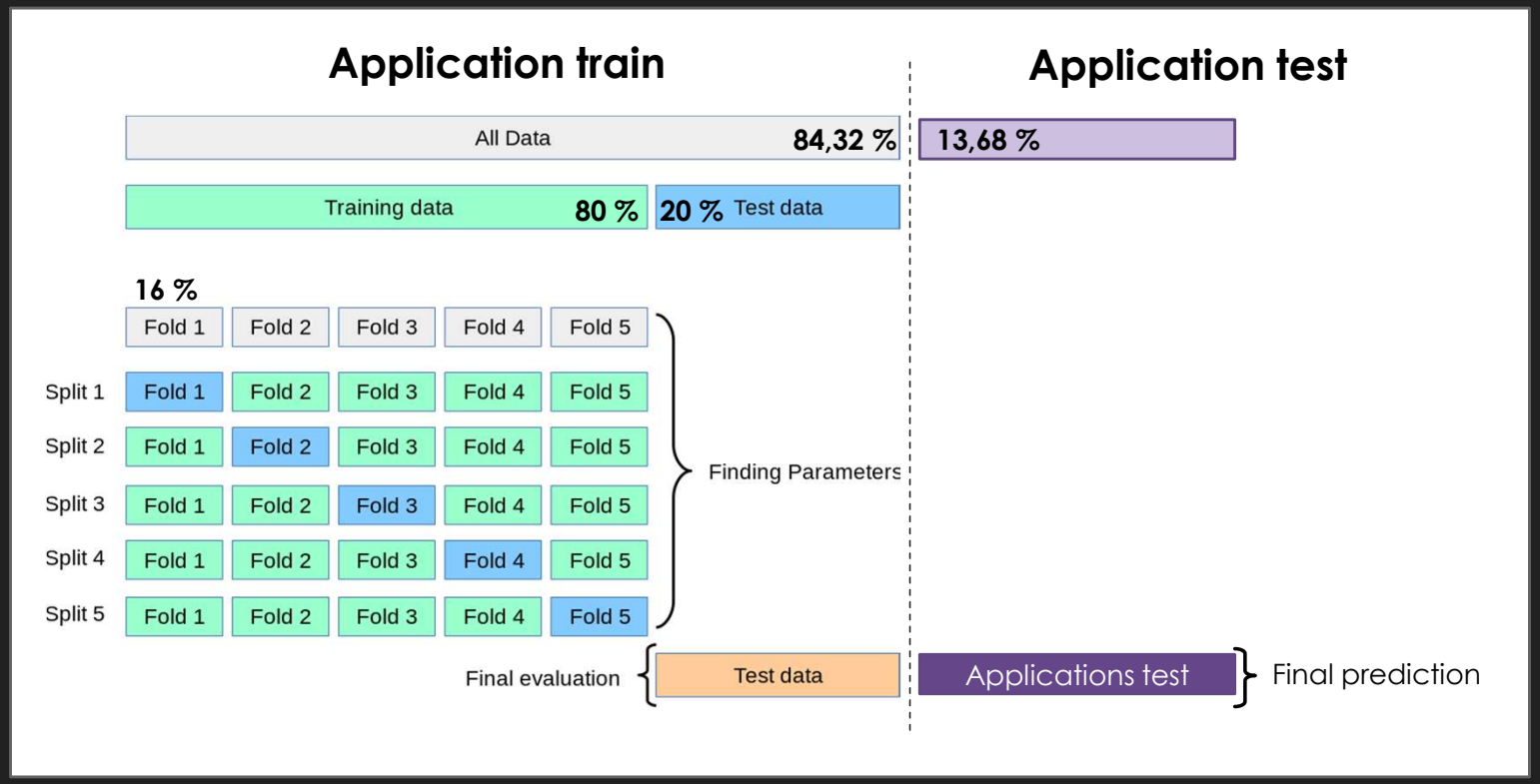
Comparison with Class 0, 1 and similar application



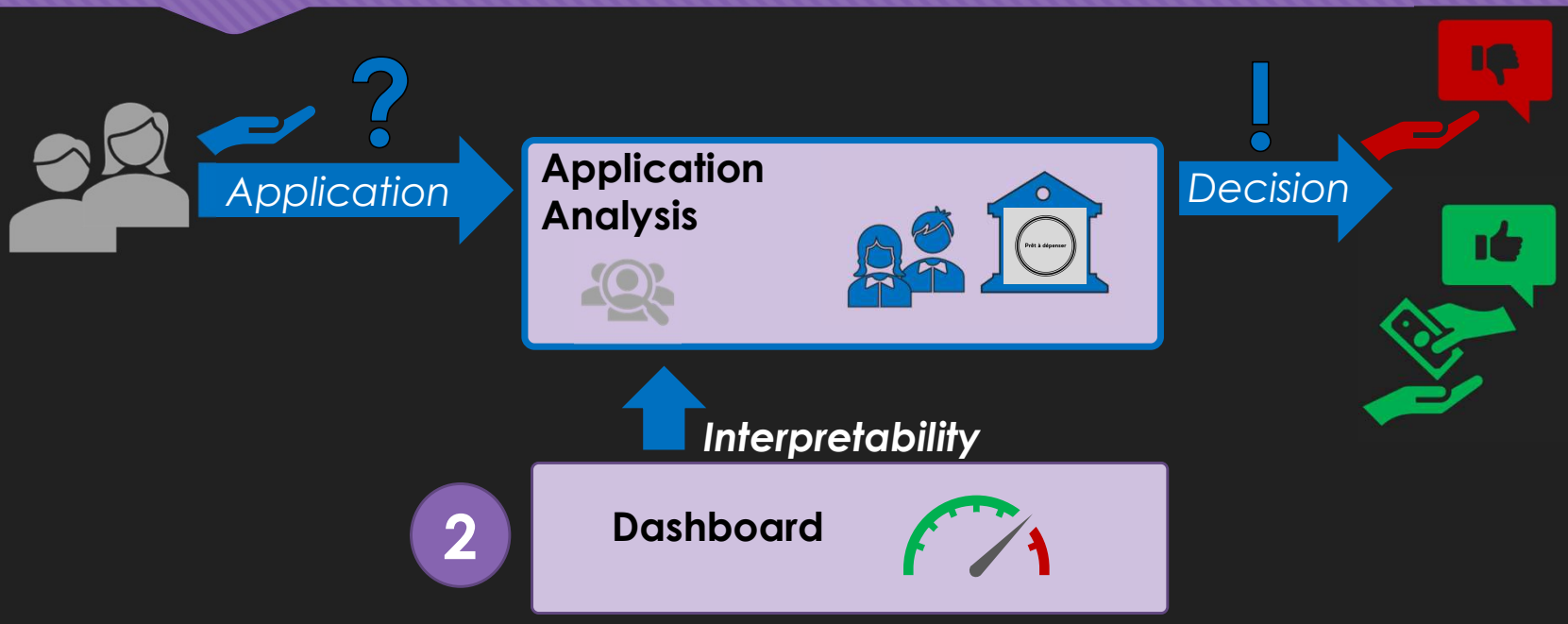
Client default rate on past instalments is **redhibitory**

Time for Demo

Q&A


Application train
Application test
All Data
84,32 %
13,68 %
Training data
80 %
20 %
Test data
16 %
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data
Applications test
Final prediction

Use Case



Done! now open the **Black Box**