03_Modelling.ipynb

September 30, 2021

1 Initialisation

[ ]:
```python
# Importations
import sys
sys.path.append('..')

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold, RepeatedStratifiedKFold
from sklearn.model_selection import cross_validate
from imblearn.pipeline import Pipeline

from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import confusion_matrix, classification_report
from imblearn.combine import SMOTETomek, SMOTEENN
from imblearn.under_sampling import TomekLinks, RandomUnderSampler
from preprocessing import preprocessor as prep
from preprocessing import preprocessor_no_scaler as prep_no_scl
from styles import *
```

[ ]:
```python
# Initialisation
train = pd.read_csv('../02_data/application_train.csv')
test = pd.read_csv('../02_data/application_test.csv')

id_error_msg = lambda x: '`SK_ID_CURR` is not unic for {} set!'.format(x)
assert len(train.SK_ID_CURR.unique()) == train.shape[0], id_error_msg('train')
assert len(test.SK_ID_CURR.unique()) == test.shape[0], id_error_msg('test')
train.set_index('SK_ID_CURR', inplace=True)
test.set_index('SK_ID_CURR', inplace=True)

print('Training set dimensions :', train.shape)

cls_size = train.TARGET.value_counts()
cls_freq = train.TARGET.value_counts(normalize=True)
```

```
print(pd.DataFrame({'size': cls_size,
                    'freq': cls_freq.apply(lambda x: '%.3f' % x)}))
```

```
Training set dimensions : (307511, 121)
     size   freq
0  282686  0.919
1   24825  0.081
```

[ ]:
```
train_sample = train[::10]
print('Sampled training set dimensions :', train_sample.shape)
cls_size = train_sample.TARGET.value_counts()
cls_freq = train_sample.TARGET.value_counts(normalize=True)
print(pd.DataFrame({'size': cls_size,
                    'freq': cls_freq.apply(lambda x: '%.3f' % x)}))
```

```
Sampled training set dimensions : (30752, 121)
    size   freq
0  28303  0.920
1   2449  0.080
```

On échantillonne le dataset en prenant 10% des points de données

[ ]:
```
X, y = train.iloc[:, 1:], train.iloc[:, 0]#.values.reshape(-1,1)
Xs, ys = train_sample.iloc[:, 1:], train_sample.iloc[:, 0]#.values.reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(Xs, ys, test_size=.2,
                                                    random_state=0)
print('X_train:', X_train.shape)
print('y_train:', y_train.shape)
print('X_test:', X_test.shape)
print('y_test:', y_test.shape)
```

```
X_train: (24601, 120)
y_train: (24601,)
X_test: (6151, 120)
y_test: (6151,)
```

2 Rééquilibrage de classes - SMOTE/Tomek

Il y a ~8% de cas de défaut dans le jeu d'entraînement contre 92% de cas sans défaut. Le déséquilibre des classes pose problème dans le cadre de la prédiction de la classe minoritaire par un algorithme de ml.

Il faut rééquilibrer les classes du jeu d'entraînement avant de sélectionner le meilleur modèle de ml

2.1 Impact de SMOTE Tomek sur la répartition des classes

```
[ ]: resamplr = SMOTETomek(tomek=TomekLinks(sampling_strategy='majority'))
     udsamplr = SMOTEENN(random_state=42)
     rusamplr = RandomUnderSampler(random_state=42)
```

```
[ ]: X_train_trans = prep.fit_transform(X_train)
     print(X_train_trans.shape)
     print(X_train_trans)
     print(y_train.shape)
     print(y_train.value_counts())
```

```
(24601, 235)
[[0.         0.09011628 0.07823375 … 1.         0.         0.        ]
 [0.         0.01162791 0.01353611 … 0.         1.         0.        ]
 [0.         0.05232558 0.15492746 … 0.         1.         0.        ]
 …
 [0.         0.14244186 0.1340753  … 0.         1.         0.        ]
 [0.1        0.12790698 0.28631022 … 0.         0.         0.        ]
 [0.3        0.06395349 0.25047455 … 0.         1.         0.        ]]
(24601,)
0    22659
1     1942
Name: TARGET, dtype: int64
```

```
[ ]: X_train_resampl, y_train_resampl = resamplr.fit_resample(X_train_trans, y_train)
     print(X_train_resampl.shape)
     print(y_train_resampl.value_counts())
```

```
(45318, 235)
0    22659
1    22659
Name: TARGET, dtype: int64
```

```
[ ]: X_train_udsampl, y_train_udsampl = udsamplr.fit_resample(X_train_trans, y_train)
     print(X_train_udsampl.shape)
     print(y_train_udsampl.value_counts())
```

```
(33702, 235)
1    22628
0    11074
Name: TARGET, dtype: int64
```

```
[ ]: X_train_rusampl, y_train_rusampl = rusamplr.fit_resample(X_train_trans, y_train)
     print(X_train_rusampl.shape)
     print(y_train_rusampl.value_counts())
```

```
(3884, 235)
```

```
0    1942
1    1942
Name: TARGET, dtype: int64
```

Rééquilibrage exécuté en 1min environ pour un jeu d'entraînement divisé par 10.

2.2 Impact de SMOTE Tomek sur l'entraînement d'un modèle

[ ]:
```
sgd = Pipeline([('p', prep), ('m', SGDClassifier())])
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
#cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=42)
scoring = ['precision_macro','recall_macro'] #, 'accuracy']
sgd_scor = cross_validate(sgd, X_train, y_train, scoring=scoring, cv=cv)
print('Model 1\n' + line_decor)
#print('accuracy scores:', sgd_scor['test_accuracy'])
print('precision scores:', sgd_scor['test_precision_macro'])
print('recall scores:', sgd_scor['test_recall_macro'])
#print('Mean Accuracy: %.4f' % np.mean(sgd_scores['test_accuracy']))
print('Mean Precision: %.4f' % np.nanmean(sgd_scor['test_precision_macro']))
print('Mean Recall: %.4f' % np.nanmean(sgd_scor['test_recall_macro']))
```

```
Model 1
--------
precision scores: [       nan 0.46056911 0.46056911 0.46056911        nan]
recall scores: [nan 0.5 0.5 0.5 nan]
Mean Precision: 0.4606
Mean Recall: 0.5000
```

Validation croisée sans SMOTE Tomek : 8.7s avec un échantillon divisé par 10

[ ]:
```
sgd_imb = Pipeline([('p', prep), ('r', resamplr), ('m', SGDClassifier())])
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
#cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=42)
scoring = ['precision_macro','recall_macro'] #, 'accuracy']
sgd_imb_scor = cross_validate(sgd_imb, X_train, y_train, scoring=scoring, cv=5)
print('Model 1 - with imbalance handling\n' + line_decor)
#print('accuracy scores:', sgd_imb_scor['test_accuracy'])
print('precision scores:', sgd_imb_scor['test_precision_macro'])
print('recall scores:', sgd_imb_scor['test_recall_macro'])
#print('Mean Accuracy: %.4f' % np.mean(sgd_imb_scores['test_accuracy']))
print('Mean Precision: %.4f' % np.nanmean(sgd_imb_scor['test_precision_macro']))
print('Mean Recall: %.4f' % np.nanmean(sgd_imb_scor['test_recall_macro']))
```

```
Model 1 - with imbalance handling
--------
precision scores: [       nan 0.55255999 0.5584412         nan 0.55571135]
recall scores: [       nan 0.66237227 0.63354292        nan 0.67955739]
Mean Precision: 0.5556
Mean Recall: 0.6585
```

Validation croisée avec SMOTE Tomek (stratégie majoritaire) : 207.6s avec un échantillon divisé par 10

```
[ ]: smote_unsmote_ratio = 207.6 / 8.7
     print('{:.2f}'.format(smote_unsmote_ratio))
```

```
23.86
```

```
[ ]: smote_unsmote_ratio = 186.5 / 9.6
     print('{:2f}'.format(smote_unsmote_ratio))
```

```
19.427083
```

Le SMOTE Tomek multiplie par un facteur 19 à 24 le temps d'exécution du modèle

Essai d'une validation croisée sans SMOTE Tomek avec tous les points du jeu d'entraînement

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
     print('X_train:', X_train.shape)
     print('y_train:', y_train.shape)
     print('X_test:', X_test.shape)
     print('y_test:', y_test.shape)
```

```
X_train: (246008, 120)
y_train: (246008, 1)
X_test: (61503, 120)
y_test: (61503, 1)
```

```
[ ]: sgd = Pipeline([('p', prep), ('m', SGDClassifier())])
     cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
     #cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=42)
     scoring = ['precision_macro','recall_macro'] #, 'accuracy']
     sgd_scor = cross_validate(sgd, X_train, y_train, scoring=scoring, cv=cv)
     print('Model 1\n' + line_decor)
     #print('accuracy scores:', sgd_scor['test_accuracy'])
     print('precision scores:', sgd_scor['test_precision_macro'])
     print('recall scores:', sgd_scor['test_recall_macro'])
     #print('Mean Accuracy: %.4f' % np.mean(sgd_scores['test_accuracy']))
     print('Mean Precision: %.4f' % np.nanmean(sgd_scor['test_precision_macro']))
     print('Mean Recall: %.4f' % np.nanmean(sgd_scor['test_recall_macro']))
```

```
Model 1
--------
precision scores: [0.45967644 0.45966627 0.45966627 0.45967562 0.45967562]
recall scores: [0.5 0.5 0.5 0.5 0.5]
Mean Precision: 0.4597
Mean Recall: 0.5000
```

Validation croisée sans SMOTE Tomek exécutée en 57.9s sur tout le jeu de données

[ ]:
```
unsampled_sampled_ratio = 57.9 / 8.7
print('{:.2f}'.format(unsampled_sampled_ratio))
```

```
6.66
```

Il faut 7 fois plus de temps pour exécuter la même chose sur 10 fois plus de données (pas parfaitement linéaire donc)

[ ]:
```
print('{:.2f}'.format(207.6 * unsampled_sampled_ratio))
```

```
1381.61
```

[ ]:
```
1381 / 60
```

[ ]:
```
23.016666666666666
```

Il faudrait 23 minutes rien que pour faire du rééquilibrage avec le jeu de données actuel. Pas souhaitable.

Il faut trouver un moyen de raccourcir le temps d'exécution du rééquilibrage.

2.3 Réduction du temps de rééquilibrage en suppprimant des colonnes

[ ]:
```
X_train_resampl_cut, y_train_resampl_cut = resamplr.fit_resample(
    X_train_trans[:,:50], y_train
    )
print(X_train_resampl_cut.shape)
print(y_train_resampl_cut.value_counts())
```

```
(45313, 50)
1    22659
0    22654
Name: TARGET, dtype: int64
```

temps d'entraînement 52s pour un jeu d'entraînement divisé par 10 avec seulement les 50 premières colonnes contre 60.5s avec toutes les colonnes.

3 Sous-échantillonage aléatoire

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
print('X_train:', X_train.shape)
print('y_train:', y_train.shape)
print('X_test:', X_test.shape)
print('y_test:', y_test.shape)
```

```
X_train: (246008, 120)
y_train: (246008,)
X_test: (61503, 120)
y_test: (61503,)
```

```
[ ]: sgd_imb = Pipeline([('p', prep), ('r', rusamplr), ('m', SGDClassifier())])
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
#cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=42)
scoring = ['precision_macro','recall_macro'] #, 'accuracy']
sgd_imb_scor = cross_validate(sgd_imb, X_train, y_train, scoring=scoring, cv=5)
print('Model 1 - with imbalance handling\n' + line_decor)
#print('accuracy scores:', sgd_imb_scor['test_accuracy'])
print('precision scores:', sgd_imb_scor['test_precision_macro'])
print('recall scores:', sgd_imb_scor['test_recall_macro'])
#print('Mean Accuracy: %.4f' % np.mean(sgd_imb_scores['test_accuracy']))
print('Mean Precision: %.4f' % np.nanmean(sgd_imb_scor['test_precision_macro']))
print('Mean Recall: %.4f' % np.nanmean(sgd_imb_scor['test_recall_macro']))
```

```
Model 1 - with imbalance handling
--------
precision scores: [0.54163367        nan 0.56050468 0.55293874        nan]
recall scores: [0.62721639        nan 0.67366715 0.67118886        nan]
Mean Precision: 0.5517
Mean Recall: 0.6574
```

4 Modèle 1 : SGD Classifier

```
[ ]: model1 = Pipeline([('p', prep), ('m', SGDClassifier())])
model1.fit(X_train, y_train)
y_pred = model1.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print('Model 1\n' + line_decor)
print('Score: %.4f' % model1.score(X_test, y_test))
print(line_decor + '\nConfusion matrix\n' + str(conf_mat))
print(classification_report(y_test, y_pred))
```

```
Model 1
--------
Score: 0.9190
--------
Confusion matrix
[[56522     0]
 [ 4981     0]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     56522
           1       0.00      0.00      0.00      4981

    accuracy                           0.92     61503
   macro avg       0.46      0.50      0.48     61503
weighted avg       0.84      0.92      0.88     61503
```

5 Modèle 2 : Random Forest Classifier

[ ]:
```python
model2 = Pipeline([('p', prep_no_scl), ('m', RandomForestClassifier())])
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scoring = ['accuracy','precision_macro','recall_macro']
scores_model2 = cross_validate(model2, X_train, y_train, scoring=scoring, cv=cv,
                               n_jobs=-1)

print('Model 2\n' + 8 * '-')
print('Mean Accuracy: %.4f' % np.mean(scores_model2['test_accuracy']))
print('Mean Precision: %.4f' % np.mean(scores_model2['test_precision_macro']))
print('Mean Recall: %.4f' % np.mean(scores_model2['test_recall_macro']))
```

[ ]:
```python
model2 = Pipeline([('p', prep_no_scl), ('m', RandomForestClassifier())])
model2.fit(X_train, y_train)
y_pred = model2.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print('Model 2\n' + 8 * '-')
print('Score: %.4f' % model2.score(X_test, y_test))
print(8 * '-' + '\nConfusion matrix\n' + str(conf_mat))
print(classification_report(y_test, y_pred))
```

```
Model 1
--------
Score: 0.9185
--------
Confusion matrix
[[56485     4]
 [ 5011     3]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     56489
           1       0.43      0.00      0.00      5014

    accuracy                           0.92     61503
   macro avg       0.67      0.50      0.48     61503
weighted avg       0.88      0.92      0.88     61503
```

[ ]:
```python
# undersmpling
# foret d'arbre -> feature importance
# lightgbm
# si besoin pca ou autre

# optimisation du threshold
# flask
```

[ ]:
```
y_pred = model2.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
```

```
[[56512     5]
 [ 4979     7]]
```

[ ]:
```
model2.get_params()
```

6 Modèle 3 : LightGBM

[ ]:
```
model3 = Pipeline([('p', prep), ('m', LGBMClassifier())])
model3.fit(X_train, y_train)
print('Score:', model3.score(X_test, y_test))
```

```
Score: 0.9192071931450498
```

[ ]:
```
y_pred = model3.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
```

```
[[56447    81]
 [ 4888    87]]
```

[ ]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     56528
           1       0.52      0.02      0.03      4975

    accuracy                           0.92     61503
   macro avg       0.72      0.51      0.50     61503
weighted avg       0.89      0.92      0.88     61503
```

[ ]:
```
# à faire

# smote tomek
# random search precision des deux classes (privilégier light_gbm)
#
# choisir optimisation recall(classe 1)
# fonction coût : manque à gagner pour chaque treshold
# treshold = + = + precision - recall
# precision élevée = on accepte tout le monde
# recall élevée = on refuse tout le monde
# regarder crer une colonne intérêts (amt credit - good price),
```

```
# optimiser mon threshold % de ça
```

7 2021-09-30 : Modélisation avec sous-échantillonage aléatoire de la classe majoriaire

[ ]:
```
# Importations
import sys
sys.path.append('..')

import pandas as pd
import numpy as np
from preprocessing import preprocessor as prep
from preprocessing import preprocessor_no_scaler as prep_no_scl
from preprocessing import CreditInfosImputer
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[ ]:
```
# Initialisation
train = pd.read_csv('../02_data/application_train.csv')
#test = pd.read_csv('../02_data/application_test.csv')

id_error_msg = lambda x: '`SK_ID_CURR` is not unic for {} set!'.format(x)
assert len(train.SK_ID_CURR.unique()) == train.shape[0], id_error_msg('train')
#assert len(test.SK_ID_CURR.unique()) == test.shape[0], id_error_msg('test')
train.set_index('SK_ID_CURR', inplace=True)
#test.set_index('SK_ID_CURR', inplace=True)

print('Training set dimensions :', train.shape)
df = train.copy()

cls_size = df.TARGET.value_counts()
cls_freq = df.TARGET.value_counts(normalize=True)
print(pd.DataFrame({'size': cls_size,
                    'freq': cls_freq.apply(lambda x: '%.3f' % x)}))
```

```
Training set dimensions : (307511, 121)
     size   freq
0  282686  0.919
1   24825  0.081
```

7.1 Test de CreditInfosImputer

7.1.1 Tout seul

[ ]:
```
credit_imputer = CreditInfosImputer()

credit_imputer.fit(df)
```

[ ]:
```
CreditInfosImputer()
```

[ ]:
```
df = train.copy()
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,1:], df.iloc[:,0],
                                                    test_size=.2)
```

[ ]:
```
credit_imputer.fit_transform(X_train, y_train)
```

[ ]:
```
           NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
SK_ID_CURR
346746             Cash loans           F            N               Y
123400             Cash loans           F            N               Y
371653             Cash loans           F            N               Y
324835             Cash loans           M            Y               Y
429236        Revolving loans           M            Y               Y
…                         …         …           …            …
447394             Cash loans           F            N               N
210991             Cash loans           M            N               N
112635             Cash loans           M            Y               Y
117429             Cash loans           F            N               N
157055             Cash loans           F            Y               N

            CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
SK_ID_CURR
346746                 0          103500.0     78192.0       6399.0
123400                 0           85500.0    314100.0      13833.0
371653                 0          247500.0   1059781.5      56592.0
324835                 0          427500.0    675000.0      49117.5
429236                 1          135000.0    270000.0      13500.0
…                    …                 …           …          …
447394                 0           81000.0    135000.0      10665.0
210991                 0          112500.0     76500.0       5670.0
112635                 0          157500.0    454500.0      23206.5
117429                 0          112500.0    296280.0      15124.5
157055                 0          270000.0    180000.0      17046.0

            AMT_GOODS_PRICE NAME_TYPE_SUITE  … FLAG_DOCUMENT_18  \
SK_ID_CURR                                   …
346746              67500.0   Unaccompanied  …                0
123400             225000.0   Unaccompanied  …                0
```

```
371653             954000.0          Family  …                0
324835             675000.0   Unaccompanied  …                0
429236             270000.0   Unaccompanied  …                0
…                         …               …  …              …
447394             135000.0          Family  …                0
210991              76500.0   Unaccompanied  …                0
112635             454500.0   Unaccompanied  …                0
117429             225000.0   Unaccompanied  …                0
157055             180000.0          Family  …                0

           FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
SK_ID_CURR
346746                    0                0                0
123400                    0                0                0
371653                    0                0                0
324835                    0                0                0
429236                    0                0                0
…                         …                …                …
447394                    0                0                0
210991                    0                0                0
112635                    0                0                0
117429                    0                0                0
157055                    0                0                0

            AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
SK_ID_CURR
346746                             0.0                        0.0
123400                             0.0                        0.0
371653                             0.0                        0.0
324835                             0.0                        0.0
429236                             0.0                        0.0
…                                    …                          …
447394                             NaN                        NaN
210991                             0.0                        0.0
112635                             0.0                        0.0
117429                             0.0                        0.0
157055                             0.0                        0.0

            AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
SK_ID_CURR
346746                             0.0                        0.0
123400                             0.0                        0.0
371653                             0.0                        0.0
324835                             0.0                        0.0
429236                             0.0                        0.0
…                                    …                          …
447394                             NaN                        NaN
```

```
210991                        0.0                        0.0
112635                        0.0                        0.0
117429                        0.0                        1.0
157055                        0.0                        0.0

            AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
SK_ID_CURR
346746                            0.0                         4.0
123400                            0.0                         0.0
371653                            1.0                         3.0
324835                            0.0                         2.0
429236                            0.0                         3.0
…                                   …                           …
447394                            NaN                         NaN
210991                            0.0                         3.0
112635                            0.0                         0.0
117429                            0.0                         4.0
157055                            0.0                         0.0

[246008 rows x 120 columns]
```

[ ]:
```
credit_imputer.fit_transform(df)
```

[ ]:
```
            TARGET NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR  \
SK_ID_CURR
100002           1         Cash loans           M            N
100003           0         Cash loans           F            N
100004           0    Revolving loans           M            Y
100006           0         Cash loans           F            N
100007           0         Cash loans           M            N
…              …                  …         …            …
456251           0         Cash loans           M            N
456252           0         Cash loans           F            N
456253           0         Cash loans           F            N
456254           1         Cash loans           F            N
456255           0         Cash loans           F            N

           FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  \
SK_ID_CURR
100002                   Y             0          202500.0    406597.5
100003                   N             0          270000.0   1293502.5
100004                   Y             0           67500.0    135000.0
100006                   Y             0          135000.0    312682.5
100007                   Y             0          121500.0    513000.0
…                      …           …                …         …
456251                   N             0          157500.0    254700.0
456252                   Y             0           72000.0    269550.0
```

```
456253                  Y             0          153000.0    677664.0
456254                  Y             0          171000.0    370107.0
456255                  N             0          157500.0    675000.0

            AMT_ANNUITY  AMT_GOODS_PRICE  … FLAG_DOCUMENT_18  \
SK_ID_CURR                                …
100002          24700.5         351000.0  …                0
100003          35698.5        1129500.0  …                0
100004           6750.0         135000.0  …                0
100006          29686.5         297000.0  …                0
100007          21865.5         513000.0  …                0
…                     …                …  …              …
456251          27558.0         225000.0  …                0
456252          12001.5         225000.0  …                0
456253          29979.0         585000.0  …                0
456254          20205.0         319500.0  …                0
456255          49117.5         675000.0  …                0

           FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
SK_ID_CURR
100002                    0                0                0
100003                    0                0                0
100004                    0                0                0
100006                    0                0                0
100007                    0                0                0
…                       …              …              …
456251                    0                0                0
456252                    0                0                0
456253                    0                0                0
456254                    0                0                0
456255                    0                0                0

           AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
SK_ID_CURR
100002                            0.0                        0.0
100003                            0.0                        0.0
100004                            0.0                        0.0
100006                            NaN                        NaN
100007                            0.0                        0.0
…                               …                        …
456251                            NaN                        NaN
456252                            NaN                        NaN
456253                            1.0                        0.0
456254                            0.0                        0.0
456255                            0.0                        0.0

            AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
```

```
SK_ID_CURR
100002                          0.0                        0.0
100003                          0.0                        0.0
100004                          0.0                        0.0
100006                          NaN                        NaN
100007                          0.0                        0.0
…                                …                        …
456251                          NaN                        NaN
456252                          NaN                        NaN
456253                          0.0                        1.0
456254                          0.0                        0.0
456255                          0.0                        2.0

            AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
SK_ID_CURR
100002                            0.0                         1.0
100003                            0.0                         0.0
100004                            0.0                         0.0
100006                            NaN                         NaN
100007                            0.0                         0.0
…                                  …                          …
456251                            NaN                         NaN
456252                            NaN                         NaN
456253                            0.0                         1.0
456254                            0.0                         0.0
456255                            0.0                         1.0

[307511 rows x 121 columns]
```

7.1.2 Dans une pipeline de prétraitements

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,1:], df.iloc[:,0],
                                                    test_size=.2)

train_prep = prep.fit_transform(X_train, y_train)
print(train_prep.shape)
#print(train_prep.shape)
```

```
(246008, 237)
```

[ ]:
```
train_prep[:5]
```

[ ]:
```
array([[0.07041798, 0.06742717, 0.05723906, …, 0.        , 0.        ,
        0.        ],
       [0.15842697, 0.2458231 , 0.15937149, …, 0.        , 0.        ,
        0.        ],
       [0.04719101, 0.03624079, 0.04826038, …, 0.        , 1.        ,
```

```
        0.        ],
       [0.12282584, 0.09124254, 0.10549944, …, 0.        , 1.        ,
        0.        ],
       [0.02247191, 0.04956125, 0.02356902, …, 0.        , 0.        ,
        0.        ]])
```

[ ]:
```
from preprocessing import get_preprocessed_set_column_names as get_feat_names

print(get_feat_names(prep))
```

```
['AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'CNT_CHILDREN',
'AMT_INCOME_TOTAL', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'OWN_CAR_AGE', 'CNT_FAM_MEMBERS',
'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CITY',
'HOUR_APPR_PROCESS_START', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
'OBS_30_CNT_SOCIAL_CIRCLE', 'DEF_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
'DAYS_LAST_PHONE_CHANGE', 'AMT_REQ_CREDIT_BUREAU_HOUR',
'AMT_REQ_CREDIT_BUREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK',
'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ_CREDIT_BUREAU_QRT',
'AMT_REQ_CREDIT_BUREAU_YEAR', 'APARTMENTS_AVG', 'BASEMENTAREA_AVG',
'YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BUILD_AVG', 'COMMONAREA_AVG',
'ELEVATORS_AVG', 'ENTRANCES_AVG', 'FLOORSMAX_AVG', 'FLOORSMIN_AVG',
'LANDAREA_AVG', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG',
'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG', 'APARTMENTS_MEDI',
'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI',
'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI',
'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_MEDI',
'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI', 'APARTMENTS_MODE',
'BASEMENTAREA_MODE', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE',
'COMMONAREA_MODE', 'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE',
'FLOORSMIN_MODE', 'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE',
'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'TOTALAREA_MODE',
'NAME_CONTRACT_TYPE', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'EMERGENCYSTATE_MODE',
'CODE_GENDER', 'WEEKDAY_APPR_PROCESS_START', 'FLAG_MOBIL', 'FLAG_EMP_PHONE',
'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL',
'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'FLAG_DOCUMENT_2',
'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6',
'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10',
'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14',
'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18',
'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20', 'FLAG_DOCUMENT_21',
'NAME_TYPE_SUITE_children', 'NAME_TYPE_SUITE_family',
'NAME_TYPE_SUITE_group_of_people', 'NAME_TYPE_SUITE_other_a',
'NAME_TYPE_SUITE_other_b', 'NAME_TYPE_SUITE_spouse_or_partner',
```

```
'NAME_TYPE_SUITE_unaccompanied', 'NAME_TYPE_SUITE_unknown',
'NAME_INCOME_TYPE_businessman', 'NAME_INCOME_TYPE_commercial_associate',
'NAME_INCOME_TYPE_maternity_leave', 'NAME_INCOME_TYPE_pensioner',
'NAME_INCOME_TYPE_state_servant', 'NAME_INCOME_TYPE_student',
'NAME_INCOME_TYPE_unemployed', 'NAME_INCOME_TYPE_working',
'NAME_EDUCATION_TYPE_academic_degree', 'NAME_EDUCATION_TYPE_higher_education',
'NAME_EDUCATION_TYPE_incomplete_higher', 'NAME_EDUCATION_TYPE_lower_secondary',
'NAME_EDUCATION_TYPE_secondary_or_secondary_special',
'NAME_FAMILY_STATUS_civil_marriage', 'NAME_FAMILY_STATUS_married',
'NAME_FAMILY_STATUS_separated', 'NAME_FAMILY_STATUS_single_or_not_married',
'NAME_FAMILY_STATUS_unknown', 'NAME_FAMILY_STATUS_widow',
'NAME_HOUSING_TYPE_coop_apartment', 'NAME_HOUSING_TYPE_house_or_apartment',
'NAME_HOUSING_TYPE_municipal_apartment', 'NAME_HOUSING_TYPE_office_apartment',
'NAME_HOUSING_TYPE_rented_apartment', 'NAME_HOUSING_TYPE_with_parents',
'OCCUPATION_TYPE_accountants', 'OCCUPATION_TYPE_cleaning_staff',
'OCCUPATION_TYPE_cooking_staff', 'OCCUPATION_TYPE_core_staff',
'OCCUPATION_TYPE_drivers', 'OCCUPATION_TYPE_high_skill_tech_staff',
'OCCUPATION_TYPE_hr_staff', 'OCCUPATION_TYPE_it_staff',
'OCCUPATION_TYPE_laborers', 'OCCUPATION_TYPE_lowskill_laborers',
'OCCUPATION_TYPE_managers', 'OCCUPATION_TYPE_medicine_staff',
'OCCUPATION_TYPE_private_service_staff', 'OCCUPATION_TYPE_realty_agents',
'OCCUPATION_TYPE_sales_staff', 'OCCUPATION_TYPE_secretaries',
'OCCUPATION_TYPE_security_staff', 'OCCUPATION_TYPE_unknown',
'OCCUPATION_TYPE_waitersorbarmen_staff', 'ORGANIZATION_TYPE_advertising',
'ORGANIZATION_TYPE_agriculture', 'ORGANIZATION_TYPE_bank',
'ORGANIZATION_TYPE_business_entity_type_1',
'ORGANIZATION_TYPE_business_entity_type_2',
'ORGANIZATION_TYPE_business_entity_type_3', 'ORGANIZATION_TYPE_cleaning',
'ORGANIZATION_TYPE_construction', 'ORGANIZATION_TYPE_culture',
'ORGANIZATION_TYPE_electricity', 'ORGANIZATION_TYPE_emergency',
'ORGANIZATION_TYPE_government', 'ORGANIZATION_TYPE_hotel',
'ORGANIZATION_TYPE_housing', 'ORGANIZATION_TYPE_industry_type_1',
'ORGANIZATION_TYPE_industry_type_10', 'ORGANIZATION_TYPE_industry_type_11',
'ORGANIZATION_TYPE_industry_type_12', 'ORGANIZATION_TYPE_industry_type_13',
'ORGANIZATION_TYPE_industry_type_2', 'ORGANIZATION_TYPE_industry_type_3',
'ORGANIZATION_TYPE_industry_type_4', 'ORGANIZATION_TYPE_industry_type_5',
'ORGANIZATION_TYPE_industry_type_6', 'ORGANIZATION_TYPE_industry_type_7',
'ORGANIZATION_TYPE_industry_type_8', 'ORGANIZATION_TYPE_industry_type_9',
'ORGANIZATION_TYPE_insurance', 'ORGANIZATION_TYPE_kindergarten',
'ORGANIZATION_TYPE_legal_services', 'ORGANIZATION_TYPE_medicine',
'ORGANIZATION_TYPE_military', 'ORGANIZATION_TYPE_mobile',
'ORGANIZATION_TYPE_other', 'ORGANIZATION_TYPE_police',
'ORGANIZATION_TYPE_postal', 'ORGANIZATION_TYPE_realtor',
'ORGANIZATION_TYPE_religion', 'ORGANIZATION_TYPE_restaurant',
'ORGANIZATION_TYPE_school', 'ORGANIZATION_TYPE_security',
'ORGANIZATION_TYPE_security_ministries', 'ORGANIZATION_TYPE_selfemployed',
'ORGANIZATION_TYPE_services', 'ORGANIZATION_TYPE_telecom',
'ORGANIZATION_TYPE_trade_type_1', 'ORGANIZATION_TYPE_trade_type_2',
```

```
'ORGANIZATION_TYPE_trade_type_3', 'ORGANIZATION_TYPE_trade_type_4',
'ORGANIZATION_TYPE_trade_type_5', 'ORGANIZATION_TYPE_trade_type_6',
'ORGANIZATION_TYPE_trade_type_7', 'ORGANIZATION_TYPE_transport_type_1',
'ORGANIZATION_TYPE_transport_type_2', 'ORGANIZATION_TYPE_transport_type_3',
'ORGANIZATION_TYPE_transport_type_4', 'ORGANIZATION_TYPE_university',
'ORGANIZATION_TYPE_xna', 'FONDKAPREMONT_MODE_not_specified',
'FONDKAPREMONT_MODE_org_spec_account', 'FONDKAPREMONT_MODE_reg_oper_account',
'FONDKAPREMONT_MODE_reg_oper_spec_account', 'FONDKAPREMONT_MODE_unknown',
'HOUSETYPE_MODE_block_of_flats', 'HOUSETYPE_MODE_specific_housing',
'HOUSETYPE_MODE_terraced_house', 'HOUSETYPE_MODE_unknown',
'WALLSMATERIAL_MODE_block', 'WALLSMATERIAL_MODE_mixed',
'WALLSMATERIAL_MODE_monolithic', 'WALLSMATERIAL_MODE_others',
'WALLSMATERIAL_MODE_panel', 'WALLSMATERIAL_MODE_stone_or_brick',
'WALLSMATERIAL_MODE_unknown', 'WALLSMATERIAL_MODE_wooden']
```

7.2 Test de Random Undersampler

[ ]:
```
rand_usampl = RandomUnderSampler()
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,1:], df.iloc[:,0],
                                                    test_size=.2)
resampling = rand_usampl.fit_resample(X_train, y_train)
```

[ ]:
```
resampling[0].shape
```

[ ]:
```
(39798, 120)
```

[ ]:
```
resampling[1].value_counts()
```

[ ]:
```
0    19899
1    19899
Name: TARGET, dtype: int64
```

7.3 Essais avec un classifieur en arbre de décision

[ ]:
```
tree_imb = Pipeline(steps=[
    ('r', rand_usampl),
    ('p', prep_no_scl),
    ('m', DecisionTreeClassifier())
    ])
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,1:], df.iloc[:,0],
                                                    test_size=.2)
```

[ ]:
```
tree_imb.fit(X_train, y_train)
```

[ ]:
```
Pipeline(steps=[('r', RandomUnderSampler()),
                ('p',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('creditinfosimputer',
                                                  CreditInfosImputer(),
                                                  ['AMT_CREDIT', 'AMT_ANNUITY',
                                                   'AMT_GOODS_PRICE']),
                                                 ('simpleimputer-1',
SimpleImputer(strategy='median'),
                                                  ['CNT_CHILDREN',
                                                   'AMT_INCOME_TOTAL',
                                                   'REGION_POPULATION_RELATIVE',
                                                   'DAYS_BIRTH',
                                                   'DAYS_EMPLOYED',
                                                   'DAYS_REGI…
FunctionTransformer(func=<function <lambda> at 0x7f15f0bb90d0>)),
                                                                 ('encoder',
OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['NAME_TYPE_SUITE',
                                                   'NAME_INCOME_TYPE',
                                                   'NAME_EDUCATION_TYPE',
                                                   'NAME_FAMILY_STATUS',
                                                   'NAME_HOUSING_TYPE',
                                                   'OCCUPATION_TYPE',
                                                   'ORGANIZATION_TYPE',
                                                   'FONDKAPREMONT_MODE',
                                                   'HOUSETYPE_MODE',
                                                   'WALLSMATERIAL_MODE'])])),
                ('m', DecisionTreeClassifier())])
```

[ ]:
```
y_pred = tree_imb.predict(X_test)
```

[ ]:
```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.94      0.59      0.72     56559
           1       0.11      0.60      0.19      4944

    accuracy                           0.59     61503
   macro avg       0.53      0.59      0.46     61503
weighted avg       0.88      0.59      0.68     61503
```

[ ]:
```
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
```

```
[[33287 23272]
 [ 1997  2947]]
```