# 02_Modelling.ipynb

September 22, 2021

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

import pandas as pd
#import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_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 preprocessing import preprocessor, preprocessor_without_scaler
```

[ ]:
```
# 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)}))

X, y = train.iloc[:, 1:], train.iloc[:, 0]
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)
```

```
Training set dimensions : (307511, 121)
     size   freq
0  282686  0.919
1   24825  0.081
X_train: (246008, 120)
y_train: (246008,)
X_test: (61503, 120)
y_test: (61503,)
```

## 1 Modèle 1 : SGD Classifier

[ ]:
```
model1 = make_pipeline(preprocessor, SGDClassifier())
model1.fit(X_train, y_train)
print('Score:', model1.score(X_test, y_test))
```

```
Score: 0.9189307838642018
```

[ ]:
```
y_pred = model1.predict(X_test)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
```

```
[[56517     0]
 [ 4986     0]]
```

## 2 Modèle 2 : Random Forest Classifier

[ ]:
```
model2 = make_pipeline(preprocessor_without_scaler, RandomForestClassifier())
model2.fit(X_train, y_train)
print('Score:', model2.score(X_test, y_test))
```

```
Score: 0.9189633026031251
```

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

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

[ ]:
```
print(X_train[:5])
```

```
           NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  \
SK_ID_CURR
320991             Cash loans           F            N               Y
258600             Cash loans           M            N               Y
```

```
316389           Cash loans          F           N               Y
239474      Revolving loans          F           N               Y
135015           Cash loans          M           N               Y

            CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT  AMT_ANNUITY  \
SK_ID_CURR
320991                 0          135000.0    189000.0       9778.5
258600                 0          112500.0    645889.5      21474.0
316389                 0           72000.0    315000.0      22954.5
239474                 0           94500.0    270000.0      13500.0
135015                 0          270000.0   1110582.0      36832.5

            AMT_GOODS_PRICE NAME_TYPE_SUITE  … FLAG_DOCUMENT_18  \
SK_ID_CURR                                   …
320991             189000.0          Family  …                0
258600             490500.0          Family  …                0
316389             315000.0   Unaccompanied  …                0
239474             270000.0   Unaccompanied  …                0
135015             909000.0   Unaccompanied  …                0

           FLAG_DOCUMENT_19 FLAG_DOCUMENT_20 FLAG_DOCUMENT_21  \
SK_ID_CURR
320991                    0                0                0
258600                    0                0                0
316389                    0                0                0
239474                    0                0                0
135015                    0                0                0

            AMT_REQ_CREDIT_BUREAU_HOUR  AMT_REQ_CREDIT_BUREAU_DAY  \
SK_ID_CURR
320991                             0.0                        0.0
258600                             0.0                        0.0
316389                             0.0                        0.0
239474                             0.0                        0.0
135015                             NaN                        NaN

            AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
SK_ID_CURR
320991                             0.0                        0.0
258600                             0.0                        0.0
316389                             0.0                        0.0
239474                             0.0                        0.0
135015                             NaN                        NaN

            AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
SK_ID_CURR
320991                            2.0                         6.0
258600                            0.0                         0.0
```

```
316389                                0.0                           3.0
239474                                0.0                           4.0
135015                                NaN                           NaN

[5 rows x 120 columns]
```

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

```
[ ]: {'memory': None,
      'steps': [('columntransformer',
        ColumnTransformer(remainder='passthrough',
                          transformers=[('simpleimputer-1',
                                         SimpleImputer(strategy='median'),
                                         ['CNT_CHILDREN', 'AMT_INCOME_TOTAL',
                                          'AMT_CREDIT', 'AMT_ANNUITY',
                                          'AMT_GOODS_PRICE',
                                          'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
                                          'DAYS_EMPLOYED', 'DAYS_REGISTRATION',
                                          'DAYS_ID_PUBLISH', 'OWN_CAR_AGE',
                                          'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
                                          'REGION_RAT…
     SimpleImputer(fill_value='Unknown',
     strategy='constant')),
                                                          ('value_formatter',
     FunctionTransformer(func=<function <lambda> at 0x7f1e9f7aa790>)),
                                                          ('encoder',
     OneHotEncoder())]),
                                         ['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'])])),
       ('randomforestclassifier', RandomForestClassifier())],
      'verbose': False,
      'columntransformer': ColumnTransformer(remainder='passthrough',
                        transformers=[('simpleimputer-1',
                                       SimpleImputer(strategy='median'),
                                       ['CNT_CHILDREN', 'AMT_INCOME_TOTAL',
                                        'AMT_CREDIT', 'AMT_ANNUITY',
                                        'AMT_GOODS_PRICE',
                                        'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
                                        'DAYS_EMPLOYED', 'DAYS_REGISTRATION',
                                        'DAYS_ID_PUBLISH', 'OWN_CAR_AGE',
                                        'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
                                        'REGION_RAT…
SimpleImputer(fill_value='Unknown',
strategy='constant')),
                                                  ('value_formatter',
FunctionTransformer(func=<function <lambda> at 0x7f1e9f7aa790>)),
                                                  ('encoder',
OneHotEncoder())]),
                                  ['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'])]),
 'randomforestclassifier': RandomForestClassifier(),
 'columntransformer__n_jobs': None,
 'columntransformer__remainder': 'passthrough',
 'columntransformer__sparse_threshold': 0.3,
 'columntransformer__transformer_weights': None,
 'columntransformer__transformers': [('simpleimputer-1',
   SimpleImputer(strategy='median'),
   ['CNT_CHILDREN',
    'AMT_INCOME_TOTAL',
    'AMT_CREDIT',
    'AMT_ANNUITY',
    'AMT_GOODS_PRICE',
    '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']),
('simpleimputer-2',
 SimpleImputer(),
 ['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']),
('simpleimputer-3',
 SimpleImputer(strategy='median'),
 ['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']),
('simpleimputer-4',
 SimpleImputer(strategy='most_frequent'),
 ['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']),
  ('pipeline-1',
   Pipeline(steps=[('nan_imputer', SimpleImputer(strategy='most_frequent')),
                   ('xna_imputer',
                    SimpleImputer(missing_values='XNA',
strategy='most_frequent')),
                   ('encoder',
                    OrdinalEncoder(categories=[['Cash loans', 'Revolving
loans'],
                                               ['N', 'Y'], ['N', 'Y'],
                                               ['No', 'Yes'], ['M', 'F'],
                                               ['MONDAY', 'TUESDAY',
'WEDNESDAY',
                                                'THURSDAY', 'FRIDAY',
'SATURDAY',
                                                'SUNDAY']]))]),
   ['NAME_CONTRACT_TYPE',
    'FLAG_OWN_CAR',
    'FLAG_OWN_REALTY',
    'EMERGENCYSTATE_MODE',
    'CODE_GENDER',
    'WEEKDAY_APPR_PROCESS_START']),
  ('pipeline-2',
   Pipeline(steps=[('imputer', SimpleImputer(strategy='most_frequent'))]),
   ['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']),
  ('pipeline-3',
   Pipeline(steps=[('imputer',
                    SimpleImputer(fill_value='Unknown', strategy='constant')),
                   ('value_formatter',
                    FunctionTransformer(func=<function <lambda> at
0x7f1e9f7aa790>)),
                   ('encoder', OneHotEncoder())]),
   ['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'])],
 'columntransformer__verbose': False,
 'columntransformer__simpleimputer-1': SimpleImputer(strategy='median'),
 'columntransformer__simpleimputer-2': SimpleImputer(),
 'columntransformer__simpleimputer-3': SimpleImputer(strategy='median'),
 'columntransformer__simpleimputer-4': SimpleImputer(strategy='most_frequent'),
 'columntransformer__pipeline-1': Pipeline(steps=[('nan_imputer',
SimpleImputer(strategy='most_frequent')),
                 ('xna_imputer',
                  SimpleImputer(missing_values='XNA',
strategy='most_frequent')),
                 ('encoder',
                  OrdinalEncoder(categories=[['Cash loans', 'Revolving loans'],
                                             ['N', 'Y'], ['N', 'Y'],
                                             ['No', 'Yes'], ['M', 'F'],
                                             ['MONDAY', 'TUESDAY', 'WEDNESDAY',
                                              'THURSDAY', 'FRIDAY', 'SATURDAY',
                                              'SUNDAY']]))]),
 'columntransformer__pipeline-2': Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent'))]),
 'columntransformer__pipeline-3': Pipeline(steps=[('imputer',
                  SimpleImputer(fill_value='Unknown', strategy='constant')),
                 ('value_formatter',
                  FunctionTransformer(func=<function <lambda> at
0x7f1e9f7aa790>)),
                 ('encoder', OneHotEncoder())]),
 'columntransformer__simpleimputer-1__add_indicator': False,
 'columntransformer__simpleimputer-1__copy': True,
 'columntransformer__simpleimputer-1__fill_value': None,
 'columntransformer__simpleimputer-1__missing_values': nan,
 'columntransformer__simpleimputer-1__strategy': 'median',
 'columntransformer__simpleimputer-1__verbose': 0,
 'columntransformer__simpleimputer-2__add_indicator': False,
 'columntransformer__simpleimputer-2__copy': True,
 'columntransformer__simpleimputer-2__fill_value': None,
 'columntransformer__simpleimputer-2__missing_values': nan,
 'columntransformer__simpleimputer-2__strategy': 'mean',
 'columntransformer__simpleimputer-2__verbose': 0,
 'columntransformer__simpleimputer-3__add_indicator': False,
 'columntransformer__simpleimputer-3__copy': True,
 'columntransformer__simpleimputer-3__fill_value': None,
 'columntransformer__simpleimputer-3__missing_values': nan,
 'columntransformer__simpleimputer-3__strategy': 'median',
 'columntransformer__simpleimputer-3__verbose': 0,
 'columntransformer__simpleimputer-4__add_indicator': False,
 'columntransformer__simpleimputer-4__copy': True,
 'columntransformer__simpleimputer-4__fill_value': None,
 'columntransformer__simpleimputer-4__missing_values': nan,
 'columntransformer__simpleimputer-4__strategy': 'most_frequent',
 'columntransformer__simpleimputer-4__verbose': 0,
 'columntransformer__pipeline-1__memory': None,
 'columntransformer__pipeline-1__steps': [('nan_imputer',
   SimpleImputer(strategy='most_frequent')),
  ('xna_imputer',
   SimpleImputer(missing_values='XNA', strategy='most_frequent')),
  ('encoder',
   OrdinalEncoder(categories=[['Cash loans', 'Revolving loans'], ['N', 'Y'],
                              ['N', 'Y'], ['No', 'Yes'], ['M', 'F'],
                              ['MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY',
                               'FRIDAY', 'SATURDAY', 'SUNDAY']]))],
 'columntransformer__pipeline-1__verbose': False,
 'columntransformer__pipeline-1__nan_imputer':
SimpleImputer(strategy='most_frequent'),
 'columntransformer__pipeline-1__xna_imputer':
SimpleImputer(missing_values='XNA', strategy='most_frequent'),
 'columntransformer__pipeline-1__encoder': OrdinalEncoder(categories=[['Cash
loans', 'Revolving loans'], ['N', 'Y'],
                            ['N', 'Y'], ['No', 'Yes'], ['M', 'F'],
                              ['MONDAY', 'TUESDAY', 'WEDNESDAY', 'THURSDAY',
                               'FRIDAY', 'SATURDAY', 'SUNDAY']]),
 'columntransformer__pipeline-1__nan_imputer__add_indicator': False,
 'columntransformer__pipeline-1__nan_imputer__copy': True,
 'columntransformer__pipeline-1__nan_imputer__fill_value': None,
 'columntransformer__pipeline-1__nan_imputer__missing_values': nan,
 'columntransformer__pipeline-1__nan_imputer__strategy': 'most_frequent',
 'columntransformer__pipeline-1__nan_imputer__verbose': 0,
 'columntransformer__pipeline-1__xna_imputer__add_indicator': False,
 'columntransformer__pipeline-1__xna_imputer__copy': True,
 'columntransformer__pipeline-1__xna_imputer__fill_value': None,
 'columntransformer__pipeline-1__xna_imputer__missing_values': 'XNA',
 'columntransformer__pipeline-1__xna_imputer__strategy': 'most_frequent',
 'columntransformer__pipeline-1__xna_imputer__verbose': 0,
 'columntransformer__pipeline-1__encoder__categories': [['Cash loans',
   'Revolving loans'],
  ['N', 'Y'],
  ['N', 'Y'],
  ['No', 'Yes'],
  ['M', 'F'],
  ['MONDAY',
   'TUESDAY',
   'WEDNESDAY',
   'THURSDAY',
   'FRIDAY',
   'SATURDAY',
   'SUNDAY']],
 'columntransformer__pipeline-1__encoder__dtype': numpy.float64,
 'columntransformer__pipeline-1__encoder__handle_unknown': 'error',
 'columntransformer__pipeline-1__encoder__unknown_value': None,
 'columntransformer__pipeline-2__memory': None,
 'columntransformer__pipeline-2__steps': [('imputer',
   SimpleImputer(strategy='most_frequent'))],
 'columntransformer__pipeline-2__verbose': False,
 'columntransformer__pipeline-2__imputer':
SimpleImputer(strategy='most_frequent'),
 'columntransformer__pipeline-2__imputer__add_indicator': False,
 'columntransformer__pipeline-2__imputer__copy': True,
 'columntransformer__pipeline-2__imputer__fill_value': None,
 'columntransformer__pipeline-2__imputer__missing_values': nan,
 'columntransformer__pipeline-2__imputer__strategy': 'most_frequent',
 'columntransformer__pipeline-2__imputer__verbose': 0,
 'columntransformer__pipeline-3__memory': None,
 'columntransformer__pipeline-3__steps': [('imputer',
   SimpleImputer(fill_value='Unknown', strategy='constant')),
  ('value_formatter',
   FunctionTransformer(func=<function <lambda> at 0x7f1e9f7aa790>)),
  ('encoder', OneHotEncoder())],
 'columntransformer__pipeline-3__verbose': False,
 'columntransformer__pipeline-3__imputer': SimpleImputer(fill_value='Unknown',
strategy='constant'),
 'columntransformer__pipeline-3__value_formatter':
FunctionTransformer(func=<function <lambda> at 0x7f1e9f7aa790>),
 'columntransformer__pipeline-3__encoder': OneHotEncoder(),
 'columntransformer__pipeline-3__imputer__add_indicator': False,
 'columntransformer__pipeline-3__imputer__copy': True,
 'columntransformer__pipeline-3__imputer__fill_value': 'Unknown',
 'columntransformer__pipeline-3__imputer__missing_values': nan,
 'columntransformer__pipeline-3__imputer__strategy': 'constant',
 'columntransformer__pipeline-3__imputer__verbose': 0,
 'columntransformer__pipeline-3__value_formatter__accept_sparse': False,
 'columntransformer__pipeline-3__value_formatter__check_inverse': True,
 'columntransformer__pipeline-3__value_formatter__func': <function
preprocessing.<lambda>(x)>,
 'columntransformer__pipeline-3__value_formatter__inv_kw_args': None,
 'columntransformer__pipeline-3__value_formatter__inverse_func': None,
 'columntransformer__pipeline-3__value_formatter__kw_args': None,
 'columntransformer__pipeline-3__value_formatter__validate': False,
 'columntransformer__pipeline-3__encoder__categories': 'auto',
 'columntransformer__pipeline-3__encoder__drop': None,
 'columntransformer__pipeline-3__encoder__dtype': numpy.float64,
 'columntransformer__pipeline-3__encoder__handle_unknown': 'error',
 'columntransformer__pipeline-3__encoder__sparse': True,
 'randomforestclassifier__bootstrap': True,
 'randomforestclassifier__ccp_alpha': 0.0,
 'randomforestclassifier__class_weight': None,
 'randomforestclassifier__criterion': 'gini',
 'randomforestclassifier__max_depth': None,
 'randomforestclassifier__max_features': 'auto',
 'randomforestclassifier__max_leaf_nodes': None,
 'randomforestclassifier__max_samples': None,
 'randomforestclassifier__min_impurity_decrease': 0.0,
 'randomforestclassifier__min_impurity_split': None,
 'randomforestclassifier__min_samples_leaf': 1,
 'randomforestclassifier__min_samples_split': 2,
 'randomforestclassifier__min_weight_fraction_leaf': 0.0,
 'randomforestclassifier__n_estimators': 100,
 'randomforestclassifier__n_jobs': None,
 'randomforestclassifier__oob_score': False,
 'randomforestclassifier__random_state': None,
 'randomforestclassifier__verbose': 0,
 'randomforestclassifier__warm_start': False}
```

## 3 Modèle 3 : LightGBM

```
[ ]: model3 = make_pipeline(preprocessor, 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
```