# NOTE MÉTHODOLOGIQUE

°°°°°o°°°°°

## A) *MÉTHODOLOGIE D'ENTRAÎNEMENT DU MODÈLE*

Nous avons 2 variables importantes à traiter dans le pré-processing à savoir :

**- les variables catégorielles et les variables numériques**

1) **variables catégorielles:** nous avons réalisé de pré-processing en utilisant la méthode One-Hot Encoding pour toutes les variables catégorielles avec plus de 2 catégories et seulement avec 2 categories via get-dummies :

**Pré-processing avec plus de 2 catégories:**

```
# One-hot encoding for categorical columns with get_dummies
def one_hot_encoder(df, nan_as_category = True):
    original_columns = list(df.columns)
    categorical_columns = [col for col in df.columns if df[col].dtype == 'object']
    df = pd.get_dummies(df, columns= categorical_columns, dummy_na= nan_as_category)
    new_columns = [c for c in df.columns if c not in original_columns]
    return df, new_columns
```

2) Création de nouvelles features exprimé en pourcentage pour capturer des effets spécifiques et améliorer les modèles telles que:
df['DAYS_EMPLOYED_PERC'] = df['DAYS_EMPLOYED'] / df['DAYS_BIRTH']
df['INCOME_CREDIT_PERC'] = df['AMT_INCOME_TOTAL'] / df['AMT_CREDIT']
df['INCOME_PER_PERSON'] = df['AMT_INCOME_TOTAL'] / df['CNT_FAM_MEMBERS']
df['ANNUITY_INCOME_PERC'] = df['AMT_ANNUITY'] / df['AMT_INCOME_TOTAL']
df['PAYMENT_RATE'] = df['AMT_ANNUITY'] / df['AMT_CREDIT']

3) Récupération de données pertinentes:

- a) Pré-processing et fusion de données issues de fichiers csv suivants :
- data application_training, data application_test, bureau et bureau_balance, previous_application, POS_CASH_balance, installements_payments, credit_card_balance

4) Enregistrement du pre-processing data dans un fichier CSV

5 Intégration et Agrégation de features numériques sur toutes les datas dans le périmètre du modèle.

6) Utilisation lightGBM

7) Utilisation de Cross-validation

8) Création de features importance globale

9) Enregistrement du modèle dans le format «pickle»

```
filename = 'finalized_model.pickle'
pickle.dump(clf, open(filename, 'wb'))
```

```
with open('finalized_model.pickle', 'rb') as handle:
    model= pickle.load(handle)
```

10) Récupération de données par utilisateur

```
donnes_utilisateur = df_api.loc[193423]
```

```
donnes_utilisateur
```

```
EXT_SOURCE_1             0.686843
EXT_SOURCE_2             0.636178
EXT_SOURCE_3             0.591977
AMT_INCOME_TOTAL    270000.000000
DAYS_BIRTH          -12478.000000
AMT_CREDIT          675000.000000
AMT_ANNUITY          34596.000000
DAYS_EMPLOYED         -456.000000
PAYMENT_RATE             0.051253
Name: 193423, dtype: float64
```

11) Prédiction de la probabilité de remboursement de crédit

```
prediction = model.predict_proba(np.array(donnes_utilisateur).reshape(1, -1)) # echantillon de 10 et 50
```

```
np.array(donnes_utilisateur) # tableau d'une liste
```

```
array([ 6.86843174e-01,  6.36177554e-01,  5.91976618e-01,  2.70000000e+05,
       -1.24780000e+04,  6.75000000e+05,  3.45960000e+04, -4.56000000e+02,
        5.12533333e-02])
```

```
np.array(donnes_utilisateur).reshape(1, -1) # tableau d'une liste de tableau
```

```
array([[ 6.86843174e-01],
       [ 6.36177554e-01],
       [ 5.91976618e-01],
       [ 2.70000000e+05],
       [-1.24780000e+04],
       [ 6.75000000e+05],
       [ 3.45960000e+04],
       [-4.56000000e+02],
       [ 5.12533333e-02]])
```

```
prediction[0] # sa première fonctionnalité EXT_SOURCE_1 a cette probabilité.
```

```
array([0.97805084, 0.02194916])
```

## *B) LA FONCTION COUT MÉTIER, ALGORITHME D'OPTIMISATION*

### ➢ Définition de l'espace de recherche des hyperparamètres

Il s'agit de définir une plage d'hyperparamètres à optimiser. Il est reconnu que la plage de paramètres étroite donne de meilleurs résultats.
Exemple: - l'optimisation des hyperparamètre avec GridSearch

```
plage_param={
    'var_smoothing': [1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14, 1e-15]
}
```

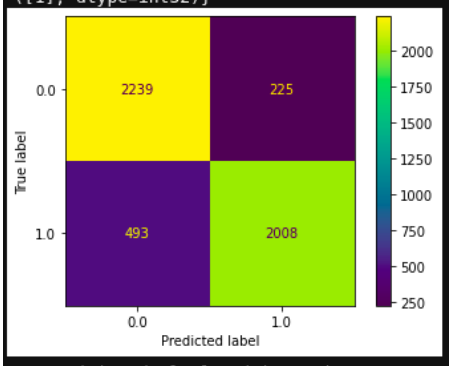
```
from sklearn.naive_bayes import GaussianNB
from sklearn.utils import check_random_state
from joblib import parallel_backend

grid = GridSearchCV(GaussianNB(), plage_param, cv=5, scoring=customer_cout_metier_scorer)
grid.fit(X_train, Y_train)
```

GridSearchCV
estimator: GaussianNB
GaussianNB

### ➢ EVALUATION DE LA PERFORMANCE DU MODÈLE

- 1) affichage de matrice de confusion du meilleur modèle retenu XGBOOST

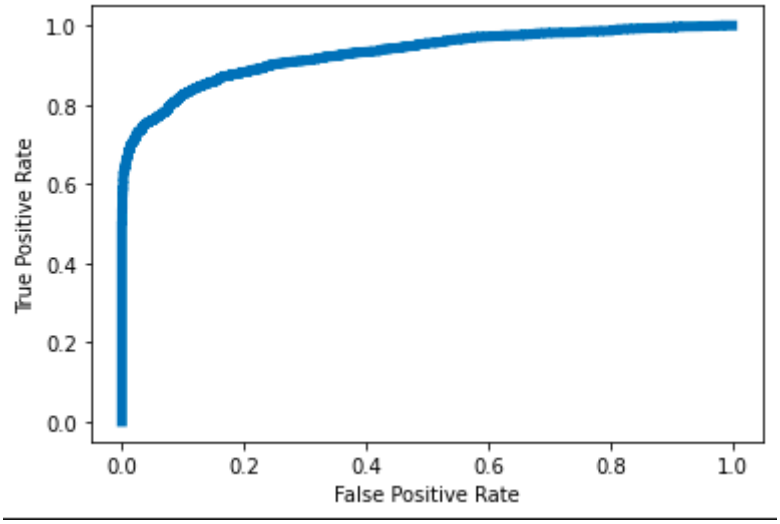
```
Training model xgb
{'mean_fit_time': array([14.05174723]), 'std_fit_time': array([1.46799307]), 'mean_score_time': array([0.02330322]), 'std_
score_time': array([0.00607867]), 'params': [{}], 'split0_test_score': array([0.93150014]), 'split1_test_score': array([0.
93008925]), 'split2_test_score': array([0.92934902]), 'split3_test_score': array([0.92397944]), 'split4_test_score': array
([0.92609438]), 'mean_test_score': array([0.92820245]), 'std_test_score': array([0.0027578]), 'rank_test_score': array
([1], dtype=int32)}
```




```
cout métier de l'algorithme xgb : 2743
```

Notre score AUC, comme indiqué sur le graphique ci-dessus est de 92,820%. Le meilleur score AUC ressorti parmi les autres modèles testés.
Du pont du vue métier, l' erreur sur les faux négatifs sont 493 clients alors qu'on pourrait accorder des crédits à ces clients. Donc, un manque à gagner pour les institutions financières.

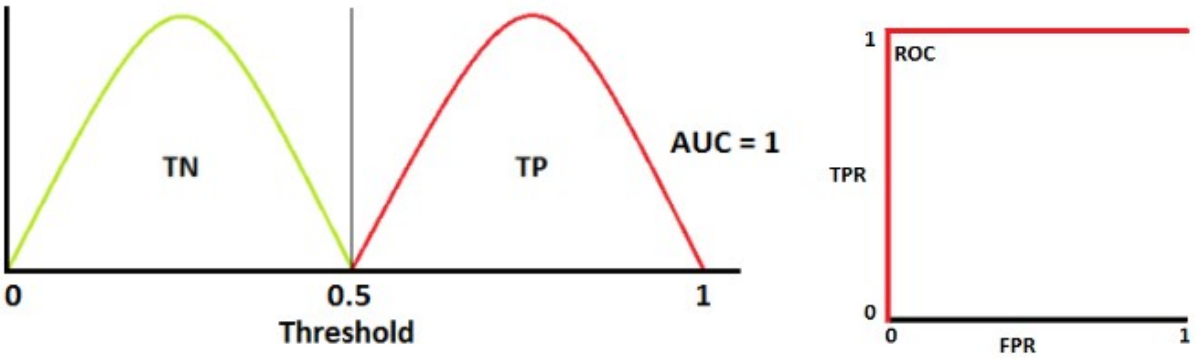
- 2) affichage et explication  AUC et ROC (métrique retenue pour la comparaison des différents modèles)



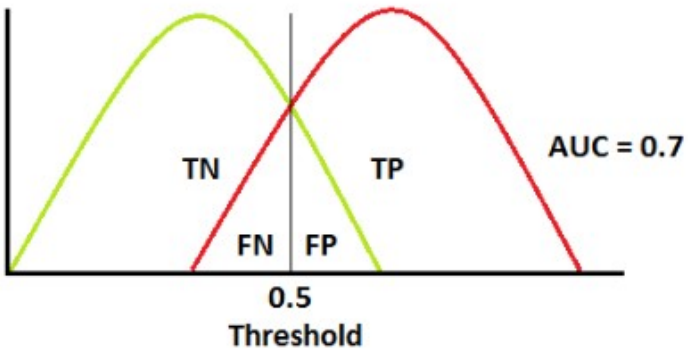

La courbe ROC est une courbe de probabilité et AUC représente le degré ou la mesure de séparabilité. Il indique à quel point le modèle est capable de faire la distinction entre les classes. Plus l'AUC est élevée, plus le modèle prédit les classes 0 comme 0 et les classes 1 comme 1.

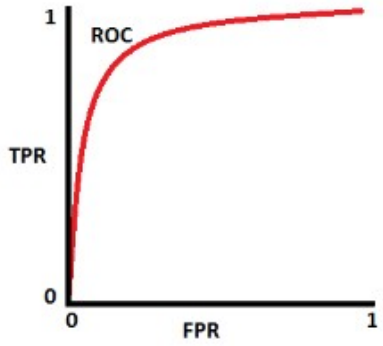
- ➔ Un excellent modèle a une AUC proche de 1, ce qui signifie qu'il a une bonne mesure de séparabilité.
- ➔ Un modèle médiocre a une AUC proche de 0, ce qui signifie qu'il a la pire mesure de séparabilité. En fait, cela signifie qu'il rend le même au résultat. Il prédit les 0 comme des 1 et les 1 comme des 0.
- ➔ Quant à l'AUC qui est de 0,5, cela signifie que le modèle n'a aucune capacité de séparation de classe.

***<u>Exemple d'illustration:</u>  Situation idéale***

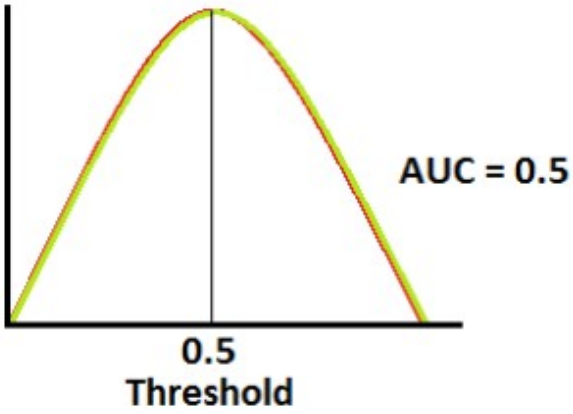


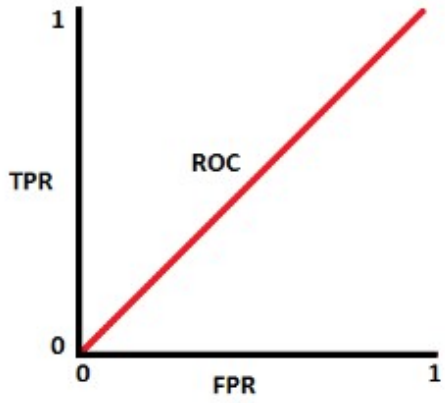

Les courbes ci-dessous montrent que les deux distributions se chevauchent, 70% de chances que le modèle puisse faire la distinction entre la classe positive et la classe négative.



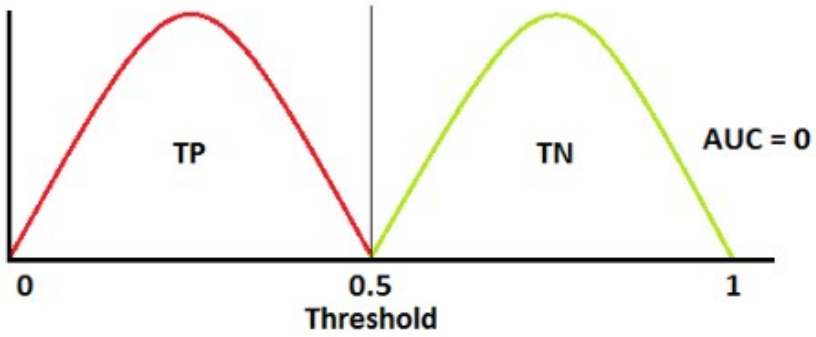



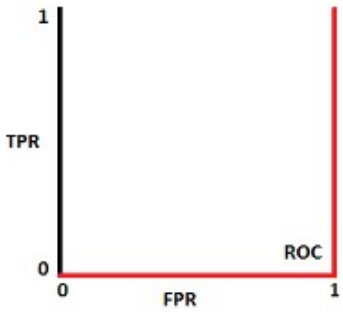

Le seuil AUC = 0,5




Cas AUC = 0,

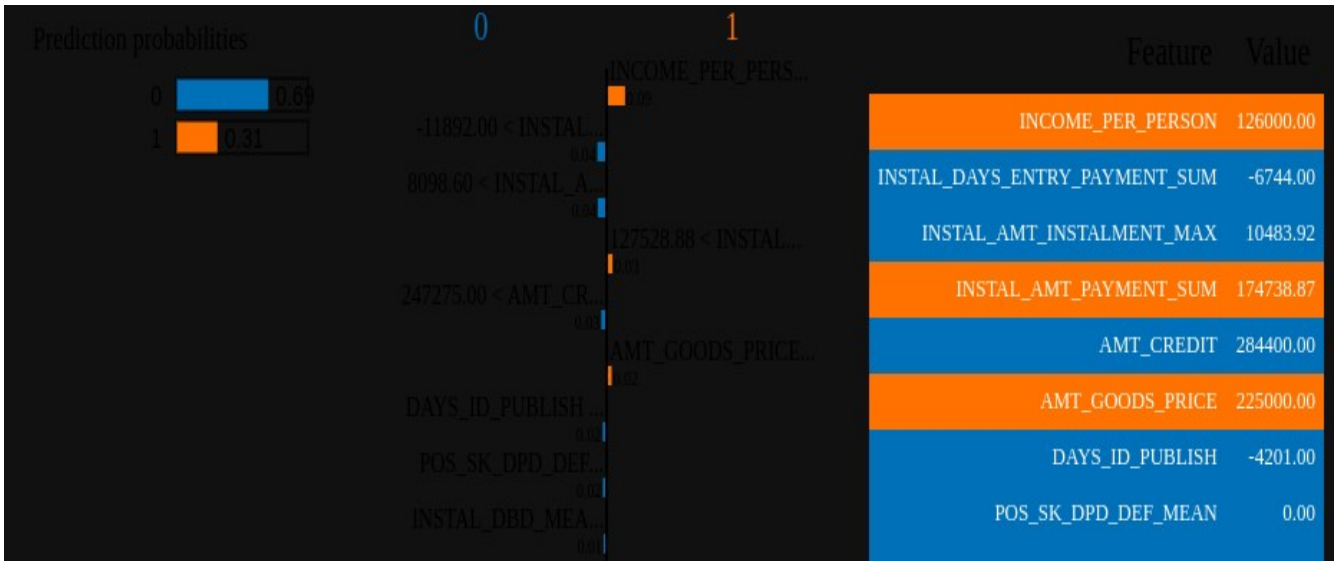



- 3) EXPLICATION du coût métier

Pour réaliser le coût métier, il faut d’abord entraîner le modèle sur différentes types d’algorithme et ensuite, on choisit le modèle d’algorithme le plus performant avec le plus minimum du coût métier possible parmi les autres.
Le metrique
L’idée est de faire en sorte qu’on cherche en permanence le minimum de coût.

## C) INTERPRETABILITE GLOBALE ET LOCALE DU MODÈLE




On constate les différentes classes de chaque feature par couleur bleue (classe 0) et orange (classe 1). Du point de vue métier et pour chaque individu, l'income_per_person de la classe 1 apporte plus d'information sur le classement s'agissant de features locales.

Les valeurs de Shap sont représentées pour chaque variable dans leur ordre d'importance, chaque point représente une valeur de Shap , pour un exemple.

Quant aux features importance globale, les features sont classées en fonction de leurs importances dans le modèle de manière générale.

## D) LES LIMITES ET LES AMELIORATIONS POSSIBLES

Les améliorations possibles sont l'optimisation des hyper-paramètres via Gridsearch CV et également Kfold pour pouvoir corriger le problème de classe déséquilibré puisque si les données sources disposent 70% de classe négative initialement, le modèle prédit ressort la majorité de cette classe lorsqu'on n'utilise pas le paramètre Kfold pour mieux stratifier les grappes du dataset.
Sur les shap, nous pouvons calculer les valeurs de Shap pour des couples de variables et étudier les interactions de variables entre elles.
Etant donné que la valeur de shap soit une approche additive, nous pouvons aisément comparer les effets des variables pour différents exemples. Effectivement, entre deux exemples qui ont des prédictions différentes, la comparaison des effets des variables peut apporter de nouvelles informations sur les variations relatives des effets.

Quant aux limites, elles se trouvent sur le test de chaque algorithme pour pouvoir fixer manuellement les entrées de chaque hyperparamètre dans l'algorithme d'optimisation de Gridsearch. On fait plusieurs tests d'entrée du paramètre jusqu'à ce que le modèle soit optimisé.