Lokman AALIOUI

Projet 7 : Implémentez un modèle de scoring

Retour de la soutenance du 28/09/2024

1ère Partie : Equilibre des classe

Dans mon approche, j'ai utilisé 3 techniques de rééchantillonnage (under-sampling, over-sampling et SMOTE), sans prendre en compte l'existence de méthodes internes aux modèles pour gérer le déséquilibre des classes.

Méthode de rééquilibrage spécifique à LGBM

LightGBM propose des paramètres spécifiques pour traiter le déséquilibre des classes sans nécessiter de rééchantillonnage des données. Deux principales méthodes sont possibles :

1) Paramètre is_unbalance

Permet à LGBM de détecter et ajuster automatiquement les déséquilibres dans les classes. Il modifie les poids des classes en fonction de leur distribution.

- LGBM (paramètre is_unbalance)

```
params = {'objective' : ['binary'],
          'n_jobs' : [-1],
          'is_unbalance' : [True],
          'num_leaves' : [20, 30, 40],
          'learning_rate' : [0.01, 0.1, 1],
          'n_estimators' : [100, 150],
          'max_depth' : [6, 15],
          'colsample_bytree': [0.8],
          'subsample' : [0.7],
          'random_state' : [42],
}

testmodel('LGBMClassifier', LGBMClassifier, par
```

30m 13.8s

Parameters (20)

Search parameters

Parameter	Value
best_learning_rate	0.01
best_max_depth	6
best_num_leaves	40
best_n_estimators	100
best_n_jobs	-1
best_objective	binary
best_random_state	42
best_subsample	0.7
cv	3
error_score	nan
estimator	LGBMClassifier()
n_jobs	-1
param_grid	{'objective': ['binary'], 'n_jobs': [-1], 'is_unbalance': [True], 'num_leaves': [20, 30, 40], 'learning_rate': [0.01, 0.1, 1], 'n_estimators': [100, 150], 'max_depth': [6, 15], 'colsample_bytree': [0.8], 'subsample': [0.7], 'random_state': [42]}
pre_dispatch	2*n_jobs
refit	accuracy
return_train_score	False
scoring	{'auc_score': make_scorer(roc_auc_score, response_method='predict'), 'accuracy': make_scorer(accuracy_score, response_method='predict')}
verbose	2

Metrics (17)

Search metrics

Metric	Value
accuracy	0.8905256654146952
best_cv_score	0.8910441951517534
confusion_matrix_fn	0.061167747914735865
confusion_matrix_fp	0.04830658667056892
confusion_matrix_tn	0.8727541745931093
confusion_matrix_tp	0.01777149082158594
f1	0.2450947415629555
metier	1.9582784579614003
recall	0.22512873326467558
roc_auc	0.7468378285198012
training_accuracy_score	0.8918653051933271
training_f1_score	0.8867107229519215
training_log_loss	0.40441419443536863
training_precision_score	0.8821394997354506
training_recall_score	0.8918653051933271
training_roc_auc	0.7638190215184127
training_score	0.8918653051933271

2) Paramètre scale_pos_weight

Ajuste manuellement l'importance relative des classes avec un ratio correspondant au déséquilibre. Le ratio choisi ici est de 11.39.

Calcul du ratio :

$$\text{scale\_pos\_weight} = \frac{\text{Nombre d'individus de la classe majoritaire}}{\text{Nombre d'individus de la classe minoritaire}}$$

$$\text{scale\_pos\_weight} = \frac{282686}{24825} \approx 11.39$$

- LGBM (paramètre scale_pos_weight)

```
params = {'objective' : ['binary'],
          'n_jobs' : [-1],
          'scale_pos_weight' : [11.39],
          'num_leaves' : [20, 30, 40],
          'learning_rate' : [0.01, 0.1, 1],
          'n_estimators' : [100, 150],
          'max_depth' : [6, 15],
          'colsample_bytree': [0.8],
          'subsample' : [0.7],
          'random_state' : [42],
}

testmodel('LGBMClassifier', LGBMClassifier, param
```

Parameters (20)

Search parameters

Parameter	Value
best_colsample_bytree	0.8
best_learning_rate	0.01
best_max_depth	6
best_num_leaves	40
best_n_estimators	100
best_n_jobs	-1
best_objective	binary
best_random_state	42
best_scale_pos_weight	11.39
best_subsample	0.7
cv	3
error_score	nan
estimator	LGBMClassifier()
n_jobs	-1
param_grid	{'objective': ['binary'], 'n_jobs': [-1], 'scale_pos_weight': [11.39], 'num_leaves': [20, 30, 40], 'learning_rate': [0.01, 0.1, 1], 'n_estimators': [100, 150], 'max_depth': [6, 15], 'colsample_bytree': [0.8], 'subsample': [0.7], 'random_state': [42]}
pre_dispatch	2*n_jobs
refit	accuracy
return_train_score	False
scoring	{'auc_score': make_scorer(roc_auc_score, response_method='predict'), 'accuracy': make_scorer(accuracy_score,

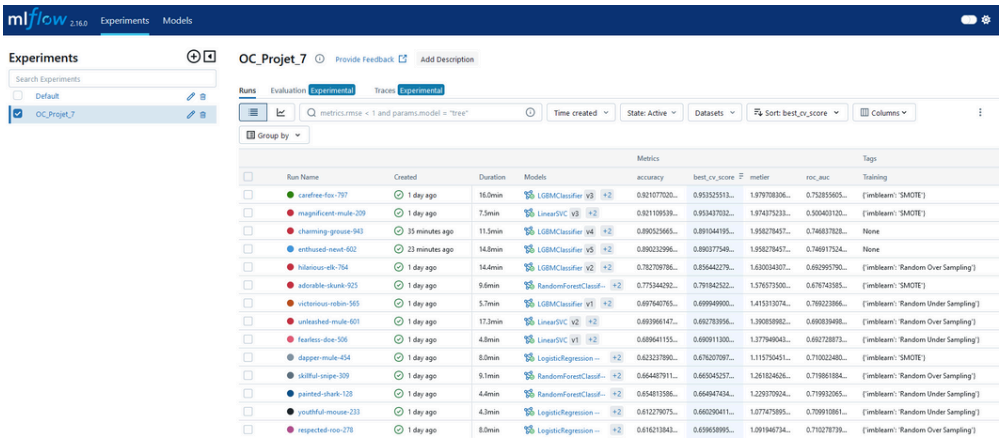
Metrics (17)

Search metrics

Metric	Value
accuracy	0.8902329967643855
best_cv_score	0.8903775499001997
confusion_matrix_fn	0.06097263548119604
confusion_matrix_fp	0.04879436775441848 4
confusion_matrix_tn	0.8722663935092597
confusion_matrix_tp	0.017966603255125765
f1	0.2466242606851914
metier	1.9582784579614003
recall	0.22760041194644695
roc_auc	0.7469175246096696
training_accuracy_score	0.8915238528828331
training_f1_score	0.8865579699952424
training_log_loss	0.40537362279308387
training_precision_score	0.8821316141179194
training_recall_score	0.8915238528828331
training_roc_auc	0.7636972767081147
training_score	0.8915238528828331

3) Comparaison avec les précédentes runs

Nous nous retrouvons avec d'excellents résultats comparés aux runs précédentes. Les 2 méthodes is_unbalance et scale_pos_weight nous donnent des metriques quasiment similaires, que ce soit l'accuracy, le best_cv_score ou le roc_auc. Néanmoins, les modèles précédemment entrainés LGBM et LinearSVC en SMOTE gardent la tête du classement avec des métrique légérement supérieures.



4) Conclusion

La facilité de mise en place de ces paramètres est un avantage indéniable quand au choix de ces derniers. Hors le fait que le rééquilibrage ayant déjà été fait pour les autres modèles, si je devais utiliser uniquement le modèle LGBM, il va sans dire que je le ferai en modifiant les paramètres proposés.

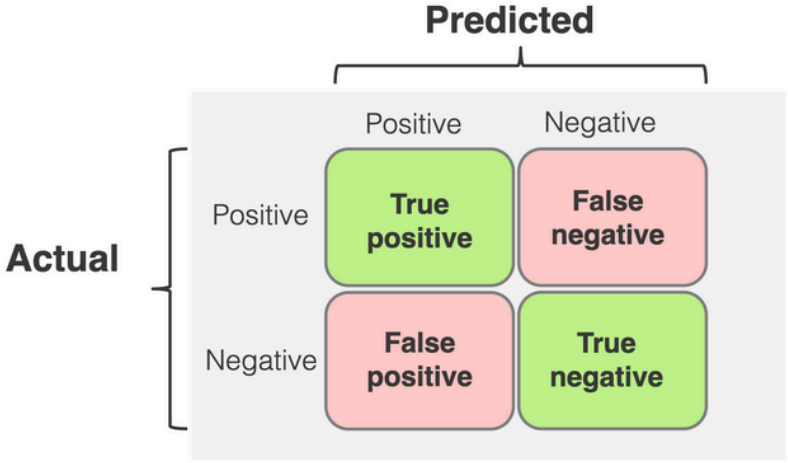
2ème Partie : Matrice de confusion

Lors de ma soutenance, j'ai présenté une matrice de confusion érronée, cela est partie d'une incompréhension de ma part. J'ai donc repris depuis le début cette notion.

1) En théorie

Dans la matrice de confusion, chaque case correspond à une comparaison entre les valeurs réelles et les valeurs prédites.

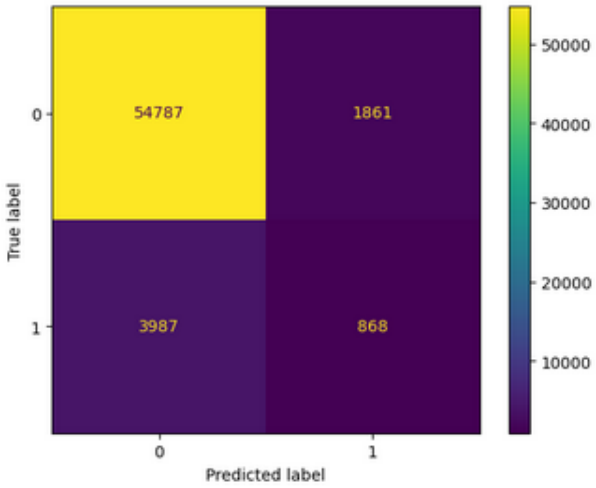
- L'axe vertical (True label) représente les classes réelles
- L'axe horizontal (Predicted label) représente les classes prédites par mon modèle



2) Analyse de la matrice de confusion

Pour rappel, dans mon dataset je dispose d'une cible qui prends les 2 valeurs suivantes :

- TARGET = 0 (92 %) représente les clients qui remboursent bien leur crédit (classe majoritaire).
- TARGET = 1 (8 %) représente les clients qui font défaut de remboursement (classe minoritaire).



- 54787 : Ce sont les vrais négatifs (TN).

Cela signifie que 54,787 clients qui remboursent bien leur crédit (classe 0) ont été correctement prédits par le modèle comme bons payeurs.

- 1861 : Ce sont les faux positifs (FP).

Cela signifie que 1,861 clients qui remboursent bien leur crédit (classe 0) ont été incorrectement prédits par le modèle comme clients à risque (classe 1). Ces individus sont donc considérés à tort comme mauvais payeurs.

- 3987 : Ce sont les faux négatifs (FN).

Cela signifie que 3,987 clients qui font défaut (classe 1) ont été incorrectement prédits comme bons payeurs (classe 0). Ces individus représentent un risque financier non identifié par le modèle.

- 868 : Ce sont les vrais positifs (TP).

Cela signifie que 868 clients qui font défaut (classe 1) ont été correctement identifiés par le modèle comme étant à risque.

3) Interprétation des resultats

Dans notre contexte de gestion du risque de non remboursement de crédit, ces chiffres impliquent :

- Vrais négatifs (TN) :

Mon modèle identifie bien la majorité des 54,787 bons payeurs, ce qui est important pour ne pas exclure de bons clients potentiels.

- Faux positifs (FP) :

1,861 bons payeurs ont été classés à tort comme mauvais payeurs. Cela pourrait causer des refus de crédit non justifiés.

- Faux négatifs (FN) :

3,987 clients risqués ont été prédits comme étant bons payeurs, ce qui est un problème majeur. Minimiser ces FN est crucial.

- Vrais positifs (TP) :

868 clients risqués ont été correctement identifiés. L'objectif est d'augmenter ce chiffre pour capter le plus de mauvais payeurs possible.

4) Conclusion

Il faudrait davantage minimiser les faux négatifs (FN) : Le modèle laisse passer environ 3,987 mauvais payeurs, ce qui représente un risque élevé pour l’entreprise. Mais aussi minimiser les faux positifs (FP) : Bien qu'il soit moins problématique de refuser du crédit à des bons clients (comparé à laisser des mauvais passer), il faudrait aussi réduire ces erreurs pour éviter des refus injustifiés.