REGAUD Agnès
Projet 7 Openclassrooms
Parcours Datascientist

Note Méthodologique

I Entraînement du modèle : étapes (2 pages maximum)

- Deux fichiers principaux au départ : un pour l'entraînement application_train et un pour la prédiction application_test
- Sur les 2 fichiers :
  - Feature engineering et préprocessing : agrégation de variables grâce à d'autres fichiers et création de nouvelles variables
  - KNN imputer pour données manquantes
  - Standardscaler pour centrer et réduire les données
- Fichier d'entraînement : Train/test split en tenant compte du déséquilibre des classes (`stratify=y`)
- Premiers essais de modèles avec :
  - Dummy Classifier (strategy=stratified)
  - Linear SVC avec sampling (`"class_weight": 'balanced'`)
  - Random Forest Classifier : 18 combinaisons d'hyper-paramètres (`"class_weight": 'balanced'`)
  - XGBoost Classifier : 12 combinaisons d'hyper-paramètres (`"scale_pos_weight" : ratio, avec ratio = 92/8 #Ratio entre les valeurs négatives et positives de la target`)

Constat : XGBoost est de loin le plus rapide et il paraît plus stable car le score métier varie de 0,56 à 0,59 et l'accuracy de 0,68 à 0,71, or Random Forest a un score métier qui varie de 0,56 à 0,81 et l'accuracy de 0,73 à 0,91. Il pourrait y avoir de l'over-fitting avec le Random Forest malgré l'utilisation de paramètres adéquats pour des données déséquilibrées (voir paragraphe suivant) : en effet l'accuracy est bonne donc l'algorithme s'entraîne bien mais il n'est pas forcément bon sur le fichier test car le score métier est trop élevé. Dans le cas du SVC, le problème se situe au niveau du temps d'entraînement qui oblige même à effectuer un sampling.
Dans les premiers essais il n'y a pas de recherche d'abaissement du score métier.

→ Choix d'utiliser XGBoost car plus stable et plus rapide, mais il faut chercher les meilleurs hyperparamètres :

- Hyperopt (Bayesian Optimization) avec XGBoost sans validation croisée et détermination des meilleurs hyper-paramètres pour un score métier à 0,56 : hyperopt est plus rapide que Gridsearchcv. Hyperopt a pour but de minimiser une objective function (ici score métier) en utilisant des calculs de probabilité qui optimisent la recherche des meilleurs hyperparamètres en se basant sur un search space d'hyperparamètres que l'on a défini. Utilisation ici d'un autre train/test split avec validation,
- GridSearchCV avec XGBoost avec validation croisée et détermination des meilleurs hyper-paramètres pour un score métier à 0,56 : en détaillant les résultats on voit que les 3 splits (on peut utiliser 5 voire 10 splits car le fichier contient suffisamment de données, mais ce serait plus long) sont homogènes pour le score métier : 0,56 ; 0,55 ; 0,57 ; pareil pour l'accuracy : 0,68 ; 0,68 ; 0,67 ; et le ROC-AUC : 0,73 ; 0 ,73 ; 0,72
- Recherche d'optimisation du score métier avec graphique de l'évolution du score en fonction de la valeur de la probabilité seuil que l'on fait varier de 0 à 1 : seuil métier
- Sauvegarde du modèle XGBoost avec les meilleurs hyper-paramètres au format .sav et avec la valeur du seuil optimal.

II Traitement du déséquilibre des classes (1 page maximum)

- Train/test split avec stratify=y
- Dummy Classifier : strategy=stratified
- Linear SVC : `"class_weight": 'balanced'`
- Random Forest Classifier : paramètre `"class_weight": 'balanced'`
- XGBoost Classifier : paramètre `"scale_pos_weight" : ratio` qui a été utilisé avec :

```
ratio = 92/8 #Ratio entre les valeurs négatives et positives de la target
```

- Hyperopt : train/test split avec validation dans lequel on a `stratify=y_train_val,` puis pour XGBoost paramètre `"scale_pos_weight" : ratio`

- GridSearchCV : XGBoost avec paramètre `scale_pos_weight = ratio`

III Fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation (1 page maximum)

- Calcul du coût métier ou score métier :

```
def score_metier(y, y_pred):
  tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()
  return round((10*fn+fp)/y.shape[0], 2)
```

On suppose ici que le coût d'un FN est 10 fois supérieur à celui d'un FP

- Optimisation du score métier :

Pour cela j'ai tracé la courbe du score métier en fonction de la probabilité seuil qui fait basculer d'une classe 0 à une classe 1. Cette probabilité varie de 0 à 1. Le minimum de cette courbe se trouve à une valeur de probabilité de 0,535 pour le fichier test. Le score métier minimum est alors de 0,55 :

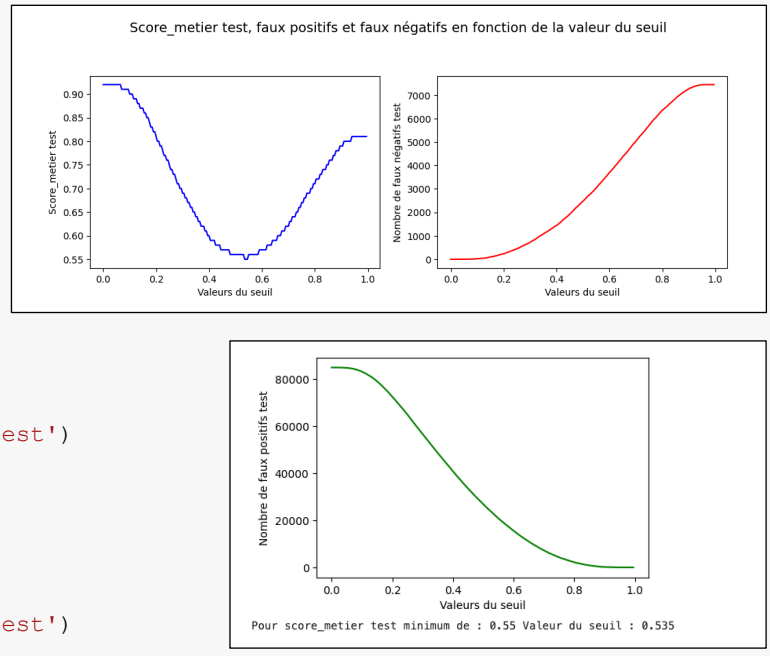
```
#Graphique évolution du score métier en fonction du seuil de predict_proba
fig = plt.figure(figsize=(12,8))
plt.suptitle("Score_metier test, faux positifs et faux négatifs en fonction de la valeur du seuil", size=14)
x=[]
y=[]
list_fn=[]
list_fp=[]
for seuil in np.arange(0,1,0.005) :
  y_test_pred = (best_model.predict_proba(X_test)[:,1] >= seuil).astype(int)
  score_train = score_metier(y_test, y_test_pred)
  tn, fp, fn, tp = confusion_matrix(y_test, y_test_pred).ravel()
  x.append(seuil)
  y.append(score_train)
  list_fn.append(fn)
  list_fp.append(fp)

plt.subplot(221)
plt.plot(x, y, color = 'blue')
plt.xlabel('Valeurs du seuil')
plt.ylabel('Score_metier test')
y = list(y)
ind = y.index(min(y))

plt.subplot(222)
plt.plot(x, list_fn, color='red')
plt.xlabel('Valeurs du seuil')
plt.ylabel('Nombre de faux négatifs test')

plt.subplot(223)
plt.plot(x, list_fp, color='green')
plt.xlabel('Valeurs du seuil')
plt.ylabel('Nombre de faux positifs test')
plt.show()
print('Pour score_metier test minimum de :', min(y), 'Valeur du seuil :', x[ind])
```

Ci-dessous les 3 graphiques :



- Evaluation des modèles :
  - ROC AUC score : Aire sous la courbe ROC. Plus ce score s'approche de 1, meilleur est le modèle.

La ROC curve est la courbe qui donne le taux de vrais positifs (fraction des positifs qui sont effectivement détectés) en fonction du taux de faux positifs(fraction des négatifs qui sont incorrectement détectés) pour un seuil donné. C'est aussi la sensibilité en fonction de 1- spécificité.

  - Accuracy : Proportion de bonnes prédictions par rapport à toutes les prédictions.

C'est le nombre de bonnes prédictions / nombre total de prédictions. Ne donne pas assez d'informations sur la performance du modèle.

IV Tableaux de synthèse des résultats (1 page maximum)

1) Dummy

Récapitulatif des résultats des essais avec DummyClassifier:

	Score métier	Accuracy	ROC AUC	f1_score	Temps d'entraînement et predict
0	0.81	0.853045	0.50103	0.081939	5.050479

2) Linear SVC

	Liste des paramètres	Score métier	Accuracy	ROC AUC	f1_score	Temps d'entraînement et predic
0	{'kernel': 'linear', 'C': 1, 'class_weight': '...	0.56	0.687175	0.673637	0.253383	1354.07295
1	{'kernel': 'linear', 'C': 2, 'class_weight': '...	0.56	0.687143	0.673681	0.253402	1910.01044
2	{'kernel': 'linear', 'C': 3, 'class_weight': '...	0.56	0.687143	0.673558	0.253324	2415.53942

3) Random Forest

Récapitulatif des résultats des essais avec RandomForestClassifier:

	Liste des paramètres	Score métier	Accuracy	ROC AUC	f1_score	Temps d'entraînement et predict
0	{'max_depth': 10, 'max_features': 'auto', 'min...	0.57	0.737233	0.666327	0.263348	82.192267
1	{'max_depth': 10, 'max_features': 'auto', 'min...	0.57	0.736626	0.666977	0.263616	81.798213
2	{'max_depth': 10, 'max_features': 'auto', 'min...	0.56	0.737190	0.667957	0.264523	79.159510
3	{'max_depth': 10, 'max_features': 'sqrt', 'min...	0.56	0.736973	0.668390	0.264764	80.134307
4	{'max_depth': 10, 'max_features': 'sqrt', 'min...	0.57	0.738068	0.666904	0.264056	81.727190
5	{'max_depth': 10, 'max_features': 'sqrt', 'min...	0.56	0.737028	0.668909	0.265160	79.315726
6	{'max_depth': 50, 'max_features': 'auto', 'min...	0.81	0.919233	0.501085	0.004808	144.841907
7	{'max_depth': 50, 'max_features': 'auto', 'min...	0.80	0.918875	0.505728	0.025267	141.316052
8	{'max_depth': 50, 'max_features': 'auto', 'min...	0.76	0.916360	0.527507	0.109624	138.577682
9	{'max_depth': 50, 'max_features': 'sqrt', 'min...	0.80	0.919385	0.501535	0.006413	148.559301
10	{'max_depth': 50, 'max_features': 'sqrt', 'min...	0.80	0.918995	0.505976	0.026065	143.205595
11	{'max_depth': 50, 'max_features': 'sqrt', 'min...	0.76	0.915873	0.527426	0.109671	142.170814
12	{'max_depth': 100, 'max_features': 'auto', 'mi...	0.81	0.919255	0.500974	0.004278	152.370180
13	{'max_depth': 100, 'max_features': 'auto', 'mi...	0.80	0.919157	0.506616	0.028400	144.264417
14	{'max_depth': 100, 'max_features': 'auto', 'mi...	0.76	0.916122	0.527132	0.108525	138.406451
15	{'max_depth': 100, 'max_features': 'sqrt', 'mi...	0.81	0.919287	0.501298	0.005609	149.346238
16	{'max_depth': 100, 'max_features': 'sqrt', 'mi...	0.80	0.919070	0.505528	0.024052	147.310698
17	{'max_depth': 100, 'max_features': 'sqrt', 'mi...	0.76	0.915959	0.527228	0.108953	135.198699

4) XGBoost

Récapitulatif des résultats des essais avec XGBoostClassifier:

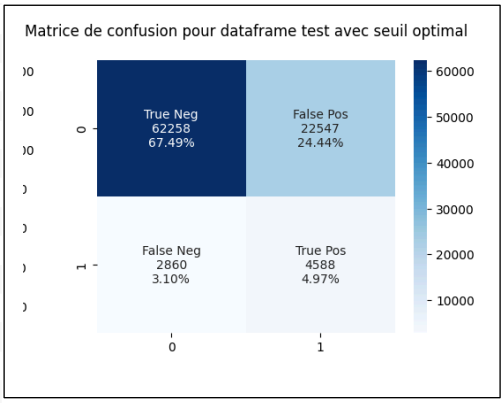
	Liste des paramètres	Score métier	Accuracy	ROC AUC	f1_score	Temps d'entraînement et predict
0	{'max_depth': 4, 'min_child_weight': 1, 'subsa...	0.57	0.684390	0.671142	0.251093	3.171784
1	{'max_depth': 4, 'min_child_weight': 1, 'subsa...	0.56	0.690471	0.673654	0.254263	1.227748
2	{'max_depth': 4, 'min_child_weight': 6, 'subsa...	0.57	0.684162	0.670773	0.250804	0.882381
3	{'max_depth': 4, 'min_child_weight': 6, 'subsa...	0.56	0.686503	0.672659	0.252591	2.371565
4	{'max_depth': 5, 'min_child_weight': 1, 'subsa...	0.58	0.692292	0.664724	0.249001	0.907949
5	{'max_depth': 5, 'min_child_weight': 1, 'subsa...	0.57	0.696790	0.669620	0.253363	0.874442
6	{'max_depth': 5, 'min_child_weight': 6, 'subsa...	0.57	0.687392	0.665366	0.248182	4.546921
7	{'max_depth': 5, 'min_child_weight': 6, 'subsa...	0.57	0.696964	0.668490	0.252673	0.904510
8	{'max_depth': 6, 'min_child_weight': 1, 'subsa...	0.59	0.702936	0.655756	0.245769	1.109402
9	{'max_depth': 6, 'min_child_weight': 1, 'subsa...	0.58	0.715760	0.659363	0.251698	5.150888
10	{'max_depth': 6, 'min_child_weight': 6, 'subsa...	0.58	0.705354	0.658540	0.248285	1.124627
11	{'max_depth': 6, 'min_child_weight': 6, 'subsa...	0.57	0.712324	0.663801	0.253789	1.070976

5) GridSearchCV

Résultats détaillés du meilleur XGBoost dans GridSearchCV : 7

mean_fit_time	0.660272
std_fit_time	0.118093
mean_score_time	0.091087
std_score_time	0.005117
param_colsample_bytree	0.6
param_gamma	0.5
param_max_depth	3
param_min_child_weight	10
param_subsample	0.8
params	{'colsample_bytree': 0.6, 'gamma': 0.5, 'max_d...
split0_test_AUC	0.732233
split1_test_AUC	0.737042
split2_test_AUC	0.727775
mean_test_AUC	0.73235
std_test_AUC	0.003784
rank_test_AUC	75
split0_train_AUC	0.760581
split1_train_AUC	0.759192
split2_train_AUC	0.762566
mean_train_AUC	0.76078
std_train_AUC	0.001385

split0_test_Score_métier	-0.56
split1_test_Score_métier	-0.55
split2_test_Score_métier	-0.57
mean_test_Score_métier	-0.56
std_test_Score_métier	0.008165
rank_test_Score_métier	1
split0_train_Score_métier	-0.53
split1_train_Score_métier	-0.53
split2_train_Score_métier	-0.53
mean_train_Score_métier	-0.53
std_train_Score_métier	0.0
split0_test_Accuracy	0.682601
split1_test_Accuracy	0.685997
split2_test_Accuracy	0.6779
mean_test_Accuracy	0.682166
std_test_Accuracy	0.00332
rank_test_Accuracy	373
split0_train_Accuracy	0.687886
split1_train_Accuracy	0.68862
split2_train_Accuracy	0.687393
mean_train_Accuracy	0.687966
std_train_Accuracy	0.000504




```
1 grid.best_params_
```

```
{'colsample_bytree': 0.6,
 'gamma': 0.5,
 'max_depth': 3,
 'min_child_weight': 10,
 'subsample': 0.8}
```

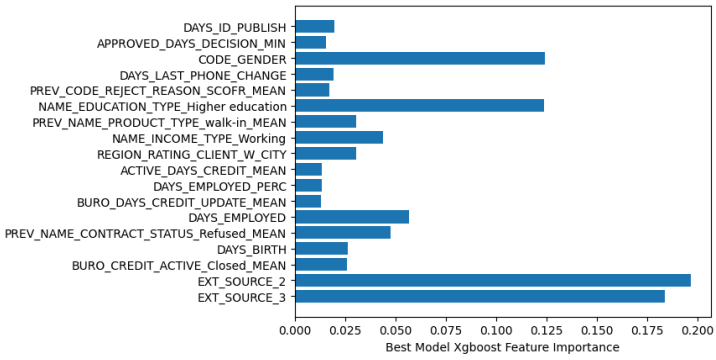
```
1 grid.best_score_
```

```
-0.56
```

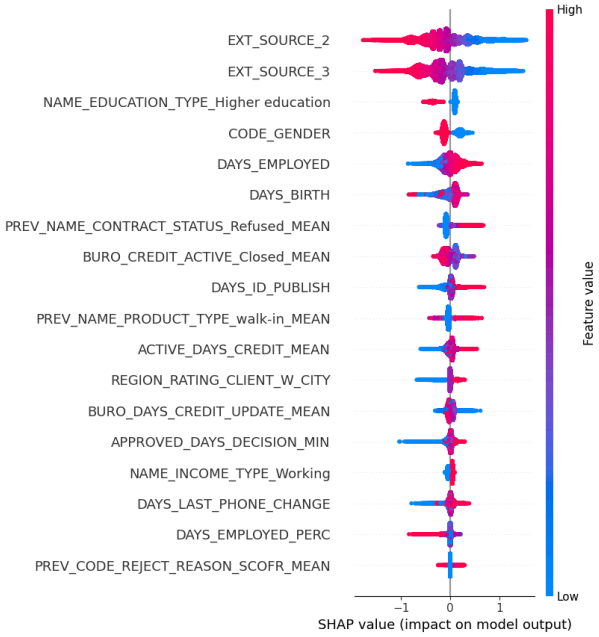
```
1 grid.best_index_
```

```
7
```

V L'interprétabilité globale et locale du modèle (1 page maximum)



Feature importance globale avec shap :



Feature importance locale sur fichier test avec shap :

12.2 Feature importance locale pour par exemple l'individu i=5 (explication pour le fichier test)

```
[61]  1 best_model.predict_proba(X_test)[5,1]
```

```
0.87853086
```

```
[62]  1 y_test_pred[5]
```

```
1
```

```
1 shap.initjs()
2
3 i = 5
4 shap_values = explainer.shap_values(X_test)
5 shap.force_plot(explainer.expected_value, shap_values[i], features=X_test[i], feature_names=list_feature_names)
```

Le graphique ci-dessus indique l'ordre d'importance des features qui entre dans le calcul de la prédiction. Ici, par exemple, EXT_SOURCE_3 participe le plus dans l'élévation de la probabilité de prédiction (en rouge). L'élévation fait tendre la prédiction vers 1, l'abaissement vers 0, et au-dessus du seuil, le client aura une réponse négative, en-dessous il aura une réponse positive. Ici la réponse est négative car il a plus de features qui élève la prédiction, en bleu il a une seule feature qui abaisse sa prédiction.

VI Les limites et les améliorations possibles (1 page maximum)

Les résultats du AUC ne sont pas assez satisfaisants, il faudrait revoir la sélection de features : les conserver toutes ou n'éliminer que celles qui sont remplies à moins de 30 ou 40% (au lieu de 60%), changer d'imputer ou encore essayer d'autres algorithmes : LightGBM par exemple.

Autres améliorations possibles :

- Pour le dashboard : changer l'intitulé des features pour qu'elles soient plus compréhensibles par le chargé de clientèle
- Faire des tests unitaires plus poussés sur l'API
- Faire des requêtes entre dashboard et API pour les features importances en plus de la seule prédiction à la demande de crédit

VII L'analyse du Data Drift (1 page maximum)

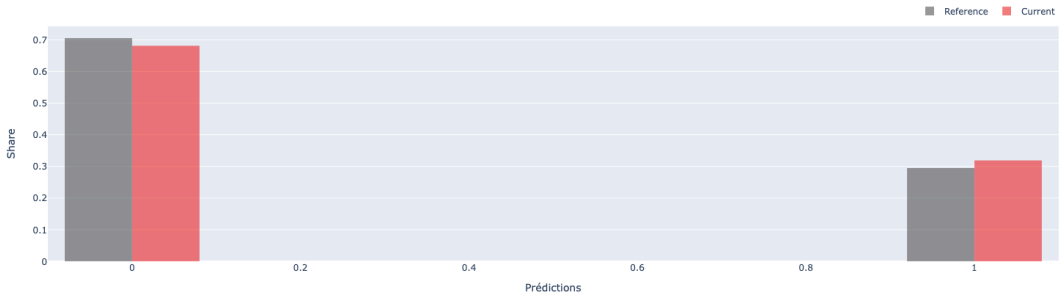
Target and data drift dashboard : pas de data drift

Drift is detected for 10.53% of features (2 out of 19). Dataset Drift is NOT detected.

Search

Feature	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
Prédictions	cat			Not Detected	Jensen-Shannon distance	0.018471
CODE_GENDER	num			Detected	Jensen-Shannon distance	0.234662
DAYS_LAST_PHONE_CHANGE	num			Detected	Wasserstein distance (normed)	0.138975
ACTIVE_DAYS_CREDIT_MEAN	num			Not Detected	Wasserstein distance (normed)	0.076276
BURO_DAYS_CREDIT_UPDATE_MEAN	num			Not Detected	Wasserstein distance (normed)	0.06949
BURO_CREDIT_ACTIVE_Closed_MEAN	num			Not Detected	Wasserstein distance (normed)	0.066807
EXT_SOURCE_3	num			Not Detected	Wasserstein distance (normed)	0.061812

Prediction Drift: not detected, drift score=0.018471 (Jensen-Shannon distance)



- NoTargetPerformance dashboard : combine différentes évaluations concernant la data quality, stability, et drift quand on a un modèle dont on connait la prédiction mais pas la target exacte car le feedback est décalé.

Ici 3 tests sur 56 ont échoué sur la qualité des données, donc rien d'alarmant.