CSC 8609 – Introduction au machine learning

Projets

Prédiction d'un défaut de paiement d'un prêt

-

Lien vers le Git du projet : MarinetHENEIN/ML-project-

27/12/2024 Marinet HENEIN

Voici les réponses aux questions du projet de machine learning :

1. Description des Données

1. **De quoi parle le jeu de données et quelle est la tâche de prédiction ?**

Le jeu de données "Give Me Some Credit" contient des informations démographiques et financières sur des clients. La tâche de prédiction consiste à déterminer si un client va être en défaut de paiement au cours des deux prochaines années (`SeriousDlqin2yrs`).

2. **Combien de variables (colonnes) contient le jeu de données ? Quels types de données contiennent-elles (catégoriques, numériques, etc.) ?**

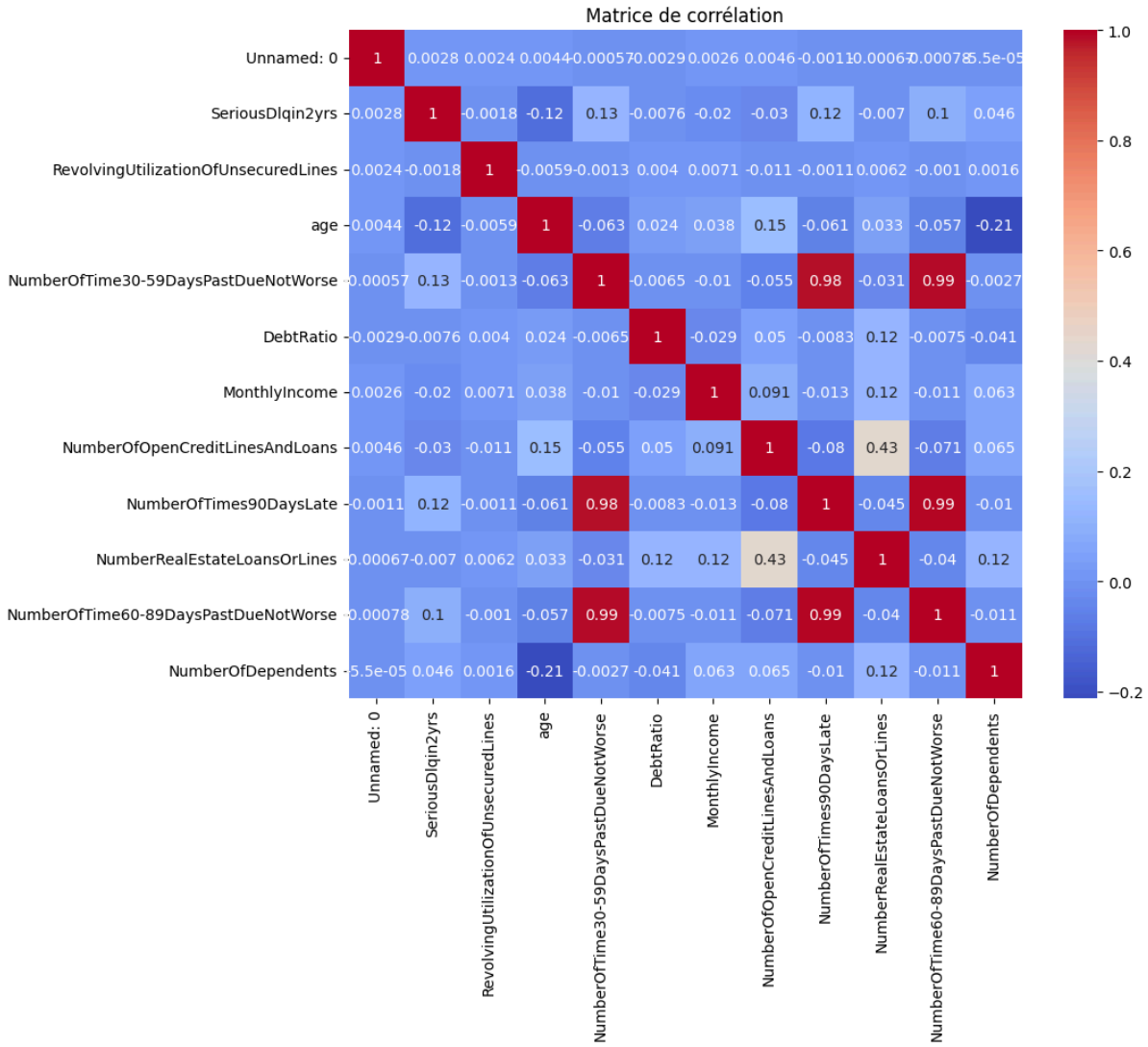
Le jeu de données contient initialement 11 colonnes. Toutes les colonnes sont numériques.

3. **Quelle est la taille du jeu de données (nombre de lignes et de colonnes) ?**

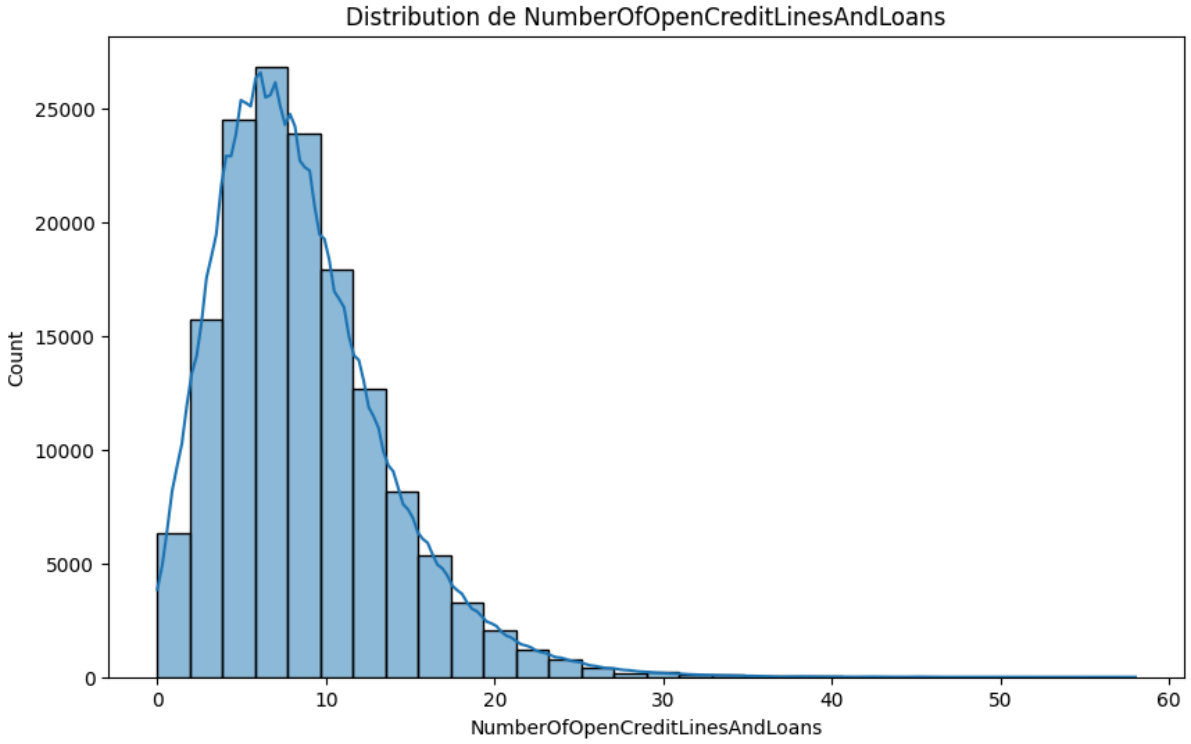
Le dataset contient 150 000 lignes et initialement 11 colonnes, ensuite réduites à 10 colonnes après suppression de la colonne `Unnamed: 0`.

4. **Y a-t-il des motifs ou des relations évidentes dans les données ? Fournissez quelques visualisations (par exemple, histogrammes, tableau de corrélation).**

Une matrice de corrélation a été visualisée pour observer les associations entre les variables :



Des histogrammes ont été réalisés pour visualiser la distribution des colonnes numériques :



5. Quels sont les défis potentiels que vous anticipez en travaillant avec ce jeu de données (par exemple, données manquantes, déséquilibre, haute dimensionnalité) ?

Certains défi risque d'être à prendre en compte lors du traitement de ces données:

- Défis incluant les valeurs manquantes dans `MonthlyIncome` et `NumberOfDependents`.

- Déséquilibre des classes (`SeriousDlqin2yrs`).

2. Nettoyage des Données

1. **Y a-t-il des valeurs manquantes dans le jeu de données ? Si oui, comment les avez-vous traitées ?**

Oui, les valeurs manquantes sont présentes dans `MonthlyIncome` et `NumberOfDependents`. Elles ont été traitées avec la médiane et zéro respectivement:

```
# 2. Nettoyage des données
data.drop(columns=['Unnamed: 0'], inplace=True, errors='ignore')
imputer = SimpleImputer(strategy="median")
data['MonthlyIncome'] = imputer.fit_transform(data[['MonthlyIncome']])
data['NumberOfDependents'] = data['NumberOfDependents'].fillna(0)
print("\nValeurs manquantes après imputation :")
print(data.isnull().sum())
```

2. **Y a-t-il des valeurs aberrantes ou des données anormales ? Comment les avez-vous identifiées et traitées ?**

Les valeurs de `RevolvingUtilizationOfUnsecuredLines` supérieures à 1 ont été considérées comme aberrantes et supprimées :

```
data = data[data['RevolvingUtilizationOfUnsecuredLines'] <= 1]
```

3. **Avez-vous rencontré des données incohérentes ou redondantes ? Comment avez-vous résolu ces problèmes ?**

La colonne `Unnamed : 0` est redondante et a été supprimée :

```
data.drop(columns=['Unnamed: 0'], inplace=True, errors='ignore')
```

4. **Avez-vous normalisé ou standardisé les données ? Expliquez pourquoi et comment.**

Oui, la normalisation a été effectuée sur les colonnes numériques pour améliorer la performance des modèles de machine learning :

```
scaler = StandardScaler()
numerical_cols = ['RevolvingUtilizationOfUnsecuredLines', 'DebtRatio', 'MonthlyIncome']
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])
print("\nDonnées après nettoyage et normalisation :")
print(data.describe())
```

5. Quels outils avez-vous utilisés pour le nettoyage des données ? Fournissez des exemples de votre code.

Utilisation de pandas et sklearn :

```
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
```

3. Ingénierie des Caractéristiques

1. **Quelles caractéristiques avez-vous sélectionnées comme les plus pertinentes pour votre modèle ? Décrivez votre processus de sélection.**

Les caractéristiques originales ont été utilisées, et une nouvelle caractéristique `IncomePerDependent` a été créée pour représenter le revenu par personne à charge.

2. **Avez-vous créé de nouvelles caractéristiques ? Si oui, expliquez ce que vous avez créé et pourquoi.**

Oui, `IncomePerDependent` a été créée pour normaliser le revenu en fonction du nombre de personnes à charge :

```
# 3. Ingénierie des caractéristiques
data['IncomePerDependent'] = data['MonthlyIncome'] / (data['NumberOfDependents'] + 1)
print("\nDonnées après ajout de nouvelles caractéristiques :")
print(data.head())
data.to_csv("engineered_data.csv", index=False)
print("Données enrichies avec nouvelles caractéristiques sauvegardées.")
```

3. **Comment avez-vous encodé les variables catégoriques (par exemple, one-hot encoding, label encoding) ? Justifiez votre choix.**

Il n'y avait pas de variables catégoriques dans ce dataset.

4. **Avez-vous transformé ou mis à l'échelle certaines caractéristiques ? Pourquoi cela était-il nécessaire pour votre projet ?**

Oui, le scaling a été appliqué aux colonne `RevolvingUtilizationOfUnsecuredLines`, `DebtRatio`, et `MonthlyIncome` pour s'assurer qu'elles ont une échelle similaire et ainsi améliorer la performance des algorithmes :

```
scaler = StandardScaler()
numerical_cols = ['RevolvingUtilizationOfUnsecuredLines', 'DebtRatio', 'MonthlyIncome']
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])
print("\nDonnées après nettoyage et normalisation :")
print(data.describe())
```

4. Sélection des Modèles

1. **Quels modèles de machine learning avez-vous testés et pourquoi avez-vous choisi ces modèles ?**

Le modèle de Random Forest a été choisi pour sa robustesse et sa capacité à gérer les données déséquilibrées.

Le modèle de Gradient Boosting (GBM) a été introduit pour comparer et évaluer ses performances.

J'en ai testé d'autres (SVM, régression logistique) mais j'ai gardé uniquement les deux ci-dessus car les résultats de performances étaient quasi similaires.

2. **Décrivez les hyperparamètres des modèles que vous avez entraînés. Quels paramètres avez-vous ajustés et comment avez-vous décidé de leurs valeurs ?**

Hyperparamètres ajustés via GridSearchCV :

- Random Forest :

```
# Modèle Random Forest
model_rf = RandomForestClassifier(random_state=42)
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 20]
}
grid_search_rf = GridSearchCV(model_rf, param_grid_rf, cv=3, scoring='roc_auc', n_jobs=-1)
grid_search_rf.fit(X_train, y_train)
best_model_rf = grid_search_rf.best_estimator_
print("Meilleurs paramètres pour Random Forest :", grid_search_rf.best_params_)
```

- Gradient Boosting :

```
# Modèle Gradient Boosting
model_gbm = GradientBoostingClassifier(random_state=42)
param_grid_gbm = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7]
}
grid_search_gbm = GridSearchCV(model_gbm, param_grid_gbm, cv=3, scoring='roc_auc', n_jobs=-1)
grid_search_gbm.fit(X_train, y_train)
best_model_gbm = grid_search_gbm.best_estimator_
print("Meilleurs paramètres pour Gradient Boosting :", grid_search_gbm.best_params_)
```

3. **Utilisez un modèle de machine learning qui n'a pas été vu dans le MOOC. Expliquez comment il fonctionne et comparez ses performances avec les autres modèles que vous avez utilisés.**

Le modèle de comparaison est donc le Gradient Boosting, (certe le concept du Boosting a été vu dans le MOOC mais c'est un des seules que j'ai testé qui à des performances meilleur que le random forest avec ce dataset, pour moi il était plus intéressant de l'étudier dans cette question).

Fonctionne en entraînant des modèles faibles en séquence, chaque nouveau modèle corrigeant les erreurs du modèle précédent.

- Performance comparée à Random Forest :

```
Rapport de classification pour Random Forest :
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     27541
           1       0.57      0.11      0.18      1795

    accuracy                           0.94     29336
   macro avg       0.76      0.55      0.58     29336
weighted avg       0.92      0.94      0.92     29336


ROC-AUC pour Random Forest : 0.853800719899094

Rapport de classification pour Gradient Boosting :
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     27541
           1       0.53      0.14      0.23      1795

    accuracy                           0.94     29336
   macro avg       0.74      0.57      0.60     29336
weighted avg       0.92      0.94      0.92     29336


ROC-AUC pour Gradient Boosting : 0.8545918017998793
```

Gradient Boosting a montré un ROC-AUC un peu meilleur que le Random Forest.

4. **Comment avez-vous géré les jeux de données déséquilibrés (si applicable) ?**

Validation croisée avec GridSearchCV pour assurer une meilleure évaluation.

5. **Comment avez-vous divisé le jeu de données en ensembles d'entraînement, de validation et de test ? Fournissez une justification pour votre choix.**

Division en 80% entraînement et 20% test.

5. Évaluation

1. **Quelles métriques avez-vous utilisées pour évaluer votre modèle (par exemple, précision, rappel, F1, RMSE) ? Pourquoi avez-vous choisi ces métriques ?**

Précision (Accuracy) : Pour mesurer la proportion de prédictions correctes.

Rappel (Recall) : Pour évaluer la capacité à identifier tous les défauts.

F1-Score : Pour un équilibre entre précision et rappel.

ROC-AUC : Pour mesurer la capacité de discrimination entre les classes, particulièrement utile pour les classes déséquilibrées.

2. **Réalisez une étude d'ablation en supprimant une ou plusieurs caractéristiques. Comment cela a-t-il affecté les performances de votre modèle ? Que révèle cela sur l'importance de ces caractéristiques ?**

Caractéristique supprimée : `IncomePerDependent`.

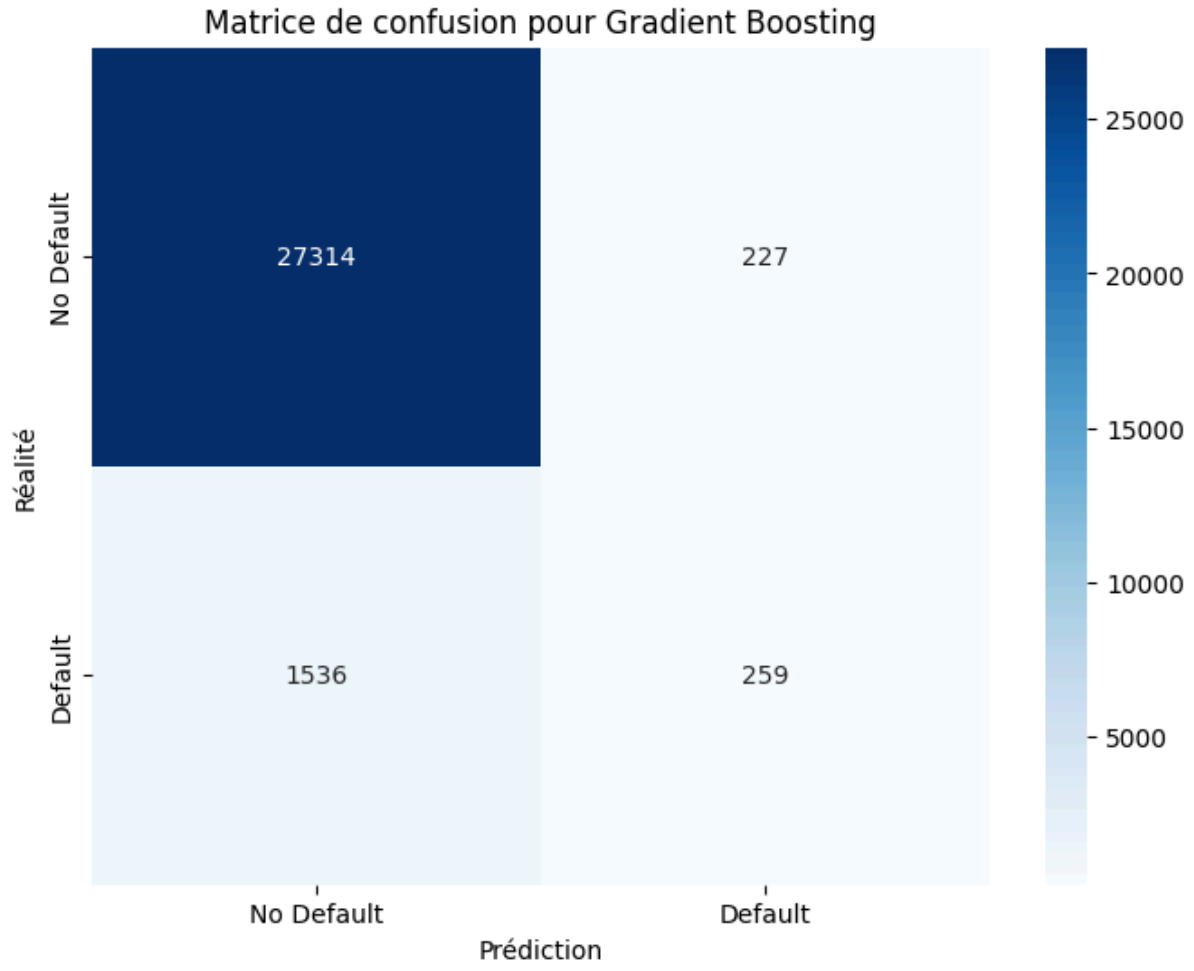
Impact sur les Performances : Diminution du ROC-AUC.

Révélation : `IncomePerDependent` est importante pour la prédiction des défauts de paiement.

```
Rapport de classification après ablation :
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     27541
           1       0.57      0.11      0.19      1795

    accuracy                           0.94     29336
   macro avg       0.76      0.55      0.58     29336
weighted avg       0.92      0.94      0.92     29336


ROC-AUC après ablation : 0.8527993867638615
```

3. **Effectuez une analyse des erreurs. Analysez les cas où le modèle a mal performé et décrivez les raisons possibles des erreurs.**

Erreur analysée via la Matrice de Confusion :



- **Faux Positifs** : Prédiction de défaut incorrect pour les clients solvables.
- **Faux Négatifs** : Non-prédiction de défaut pour les clients insolvables.

Raisons possibles : Variabilité imprévisible des comportements financiers des clients.

4. **Avez-vous effectué une validation croisée ? Si oui, quelle méthode avez-vous utilisée et quelles informations en avez-vous tirées ?**

Validation Croisée Utilisée : GridSearchCV.

Informations Tirées : Détermination des meilleurs hyperparamètres et évaluation robuste malgré le déséquilibre des classes.

```
Meilleurs paramètres pour Random Forest : {'max_depth': 10, 'n_estimators': 200}
Meilleurs paramètres pour Gradient Boosting : {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200}
```

5. **Comparez les performances de votre modèle et pipeline avec un ou plusieurs modèles de base (dont le modèle aléatoire). Quelle était ces modèles de base et combien votre modèle a-t-il amélioré les résultats ?**

Modèle de Base : Random Forest.

- **ROC-AUC pour Random Forest** : 0,854

Autre modèle :

Gradient Boosting.

- **ROC-AUC pour Gradient Boosting** : 0,855

[Autres tests]

Régression Logistique.

- **ROC-AUC pour Régression Logistique :** 0.847.

SVM.

- **ROC-AUC pour SVM :** 0.849.

6. Discussion et Conclusion

1. **Quels ont été les défis les plus importants que vous avez rencontrés pendant le projet et comment les avez-vous surmontés ?**

Les défis que j'ai rencontrés étaient surtout sur le fait de faire fonctionner chacune des parties du code et en changeant de modèle à chaque fois pour faire les tests et sélectionner ceux qui sont intéressants.

2. **Sur la base de vos résultats, à quel point votre modèle est-il fiable pour la tâche de prédiction ? Quelles limitations présente-t-il ?**

Random Forest et Gradient Boosting, ont montré une bonne performance avec des scores ROC-AUC supérieurs à 0,85. Certaines limitations peuvent se faire ressentir grâce à la matrice de confusion qui démontre la présence de faux positifs/faux négatifs .

3. **Si vous aviez plus de temps ou de ressources, quelles étapes supplémentaires prendriez-vous pour améliorer votre modèle ?**

Avec plus de temps, peut-être chercher d'autres modèles que ceux qu'on a vu au-dessus ou chercher à améliorer/ajuster les hyperparamètres avec d'autres méthodes que le GridSearch.

4. **Comment les résultats de votre projet se comparent-ils à vos attentes (par exemple, basées sur la littérature, l'intuition ou la connaissance du domaine) ?**

Les résultats sont en accord avec les notions du cours qu'on a vu tout le long du module avec le MOOC, chacune des étapes qu'on a réalisé ici depuis le nettoyage de la données jusqu'à l'évaluation des performances est développée dans le MOOC.

Je m'attendais peut-être pas à ses résultats là en termes de performances, l'écart est vraiment très fin entre chaque modèle.

5. **Résumez les leçons les plus importantes que vous avez apprises en réalisant ce projet.**

Ce projet m'a permis de mettre de bout en bout chacune des étapes du traitement de la données jusqu'au choix et l'entraînement des modèles, puis l'évaluation de ceux-ci. Pour ça il faut y aller vraiment pas à pas et de manière claire dans les étapes (documenter/commenter les étapes) sinon les erreurs s'accumulent et on s'y perd facilement.