Prédiction de défaut de prêt

Projet Fouille de Données

TONG Boan WANG Wenchi

Plan

- Introduction
- Traitement des données
- Modèle
- Résultat et discussion

Introduction

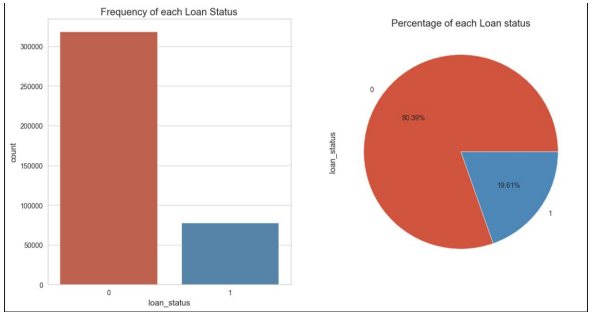
- Dataset source : https://www.kaggle.com/datasets/jeandedieunyandwi/lending-club-dataset
- Chaque ligne de dataset comporte 26 features et un target feature.
- Target feature : loan_status (0 : remboursement l'intégralité du prêt, 1 : défaut de paiement)
- But : construction d'un modèle de prédiction de défaut de paiement des prêts à partir d'informations sur les 400K demandeurs de prêts.

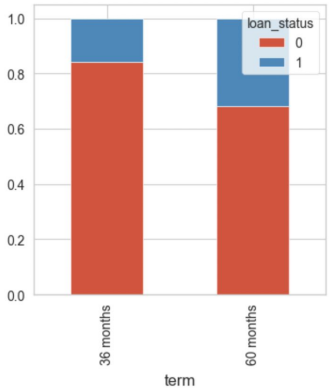
Traitement des données

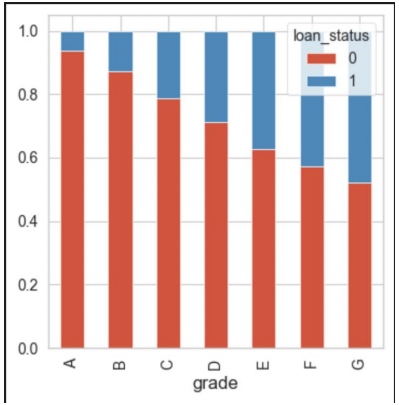
- Analyse des données
- Traitement des valeurs manquantes et des valeurs aberrantes
- feature engineering
- Train-Test-Validation Split

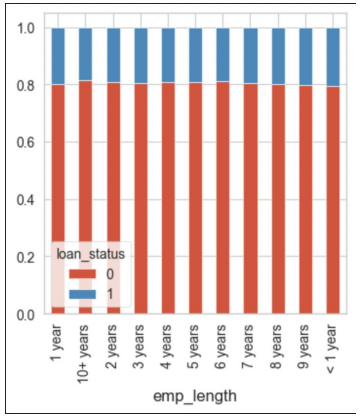
Analyse des données :

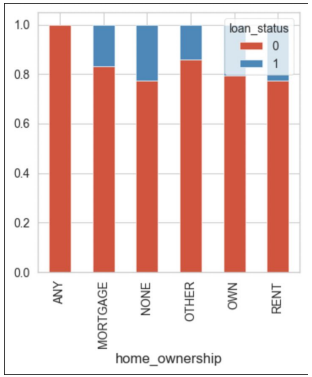
Numerical Features / Categorical Features

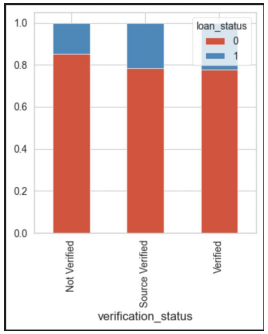


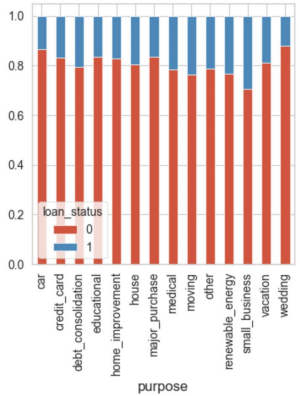


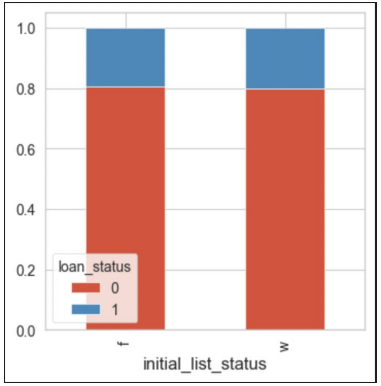


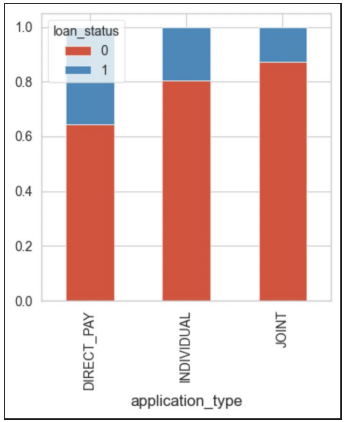






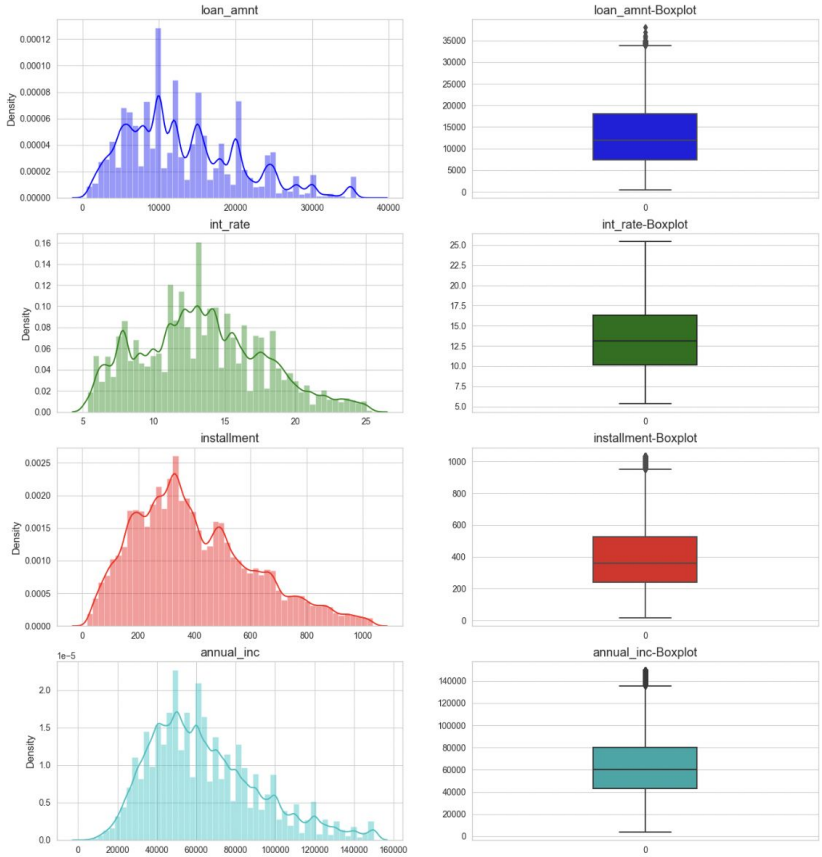


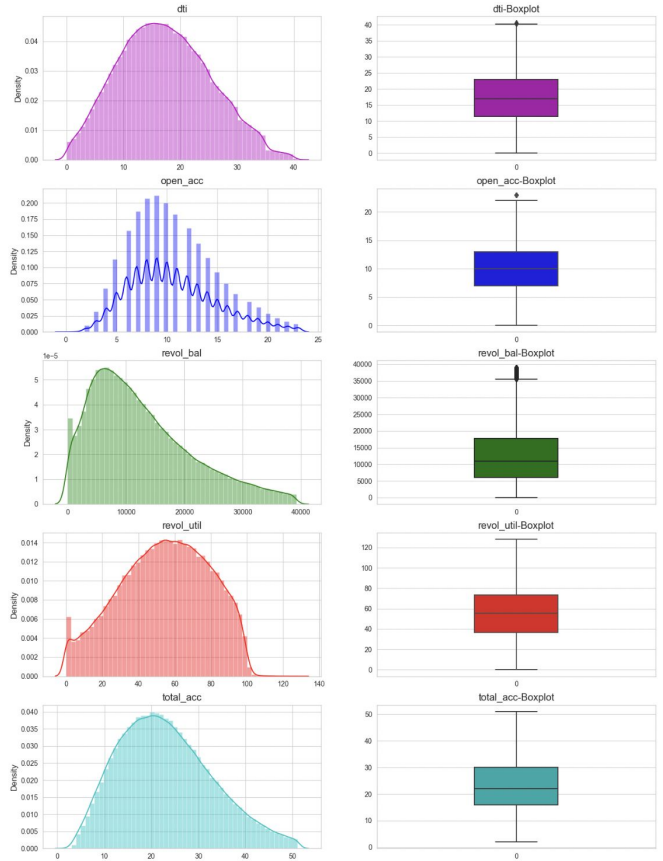




Analyse des données :

Numerical Features après Traitement des données aberrantes





Traitement des valeurs

Les valeurs manquantes:

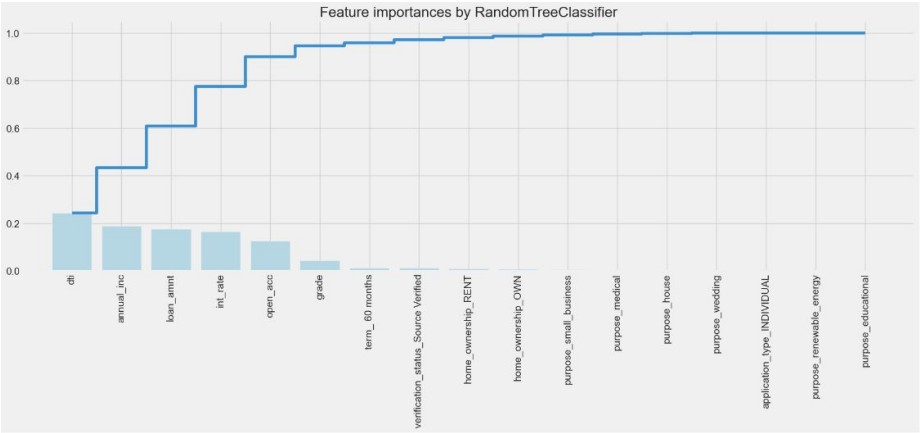
mort_acc	0.09543	-> mode
emp_title	0.05789	-> drop
emp_length	0.04621	-> drop
title	0.00443	-> drop
pub_rec_bankruptcies	0.00135	-> median
revol_util	0.00070	-> median

Les valeurs aberrantes:

Remplacer par un médian.

Feature engineering

- **Feature abstraction:**
  - Variables ordonnées à valeurs multiples (‘grade’) :
    Créer d'abord un mapping, puis utiliser la fonction replace( ) de pandas pour convertir le mapping.
  - Variables non ordonnées à valeurs multiples ('term', 'home_ownership', 'verification_status','purpose','application_type','initial_list_status') :
    Utiliser la méthode get_dummies( ) de pandas pour créer des features virtuelles pour one-hot encoding.
- **Feature scaling:**
  - StandardScaler
- **Feature selection:**
  - wrapper approach :
    Sélection de 20 features présentant la plus forte corrélation avec le target feature par la méthode Recursive Feature Elimination.
  - filter approach :
    Utilisez le heatmap pour trouver les features redondantes et les supprimer.
  - embedded approach :
    Déterminer l'importance des features avec l'algorithme Random Forest.



Resultat : (9 features)

'loan_amnt', 'grade', 'annual_inc', 'dti', 'open_acc','term_ 60 months', 'home_ownership_OWN', 'home_ownership_RENT','verification_status_Source Verified'

Modèle : Logistic Regression

Optimiser les paramètres par GridSearchCV :

```
param_grid = {'C': [0.01,0.1, 1, 10, 100, 1000]}

kflod = StratifiedKFold(n_splits=10, shuffle = True,random_state=0)
model = LogisticRegression(class_weight='balanced')

grid_search = GridSearchCV(model,param_grid, cv= kflod)
grid_search.fit(X_train_new, y_train_new)
```

Best parameters: {'C': 0.01}

Best cross-validation score: 0.66246

Modèle : Logistic Regression

Les Paramètres avec C=0.01
class_weight='balanced'

	precision	recall	f1-score	support
0	0.88	0.67	0.76	95547
1	0.32	0.63	0.42	23262
accuracy			0.66	118809
macro avg	0.60	0.65	0.59	118809
weighted avg	0.77	0.66	0.69	118809

Test set accuracy score: 0.66184

Area under the ROC curve : 0.650410

Les Paramètres avec C=0.01 , et après SMOTE

	precision	recall	f1-score	support
0	0.88	0.67	0.76	95547
1	0.32	0.63	0.42	23262
accuracy			0.66	118809
macro avg	0.60	0.65	0.59	118809
weighted avg	0.77	0.66	0.69	118809

Test set accuracy score: 0.66157

Area under the ROC curve : 0.650747

Modèle : Random Forest Classifier

Le resultat du classificateur forêt aléatoire sur l'ensemble de test (Utiliser les paramètres par défaut)

	precision	recall	f1-score	support
0	0.83	0.90	0.87	95547
1	0.40	0.26	0.32	23262
accuracy			0.78	118809
macro avg	0.62	0.58	0.59	118809
weighted avg	0.75	0.78	0.76	118809

Modèle : Random Forest Classifier

Optimiser les paramètres par GridSearchCV :

```
num_estimator = {'n_estimators':range(50,300,50)}
gs1 = GridSearchCV(estimator = rf,param_grid = num_estimator,scoring='roc_auc',cv = 5)
gs1.fit(X_train_rf,y_train_rf)
print(gs1.best_estimator_)
print(gs1.best_score_)
```

✓ 58m 58.3s Python

```
RandomForestClassifier(n_estimators=250)
0.9496932285036467
```

Les Paramètres sélectionnés : max_depth=9, min_samples_split=12, n_estimators=250

Modèle : Random Forest Classifier

Optimiser les paramètres par BayesianOptimization :

```
from bayes_opt import BayesianOptimization
def RF_evaluate(n_estimators, min_samples_split, max_features, max_depth):
    val = cross_val_score(
            RandomForestClassifier(n_estimators=int(n_estimators),
                        min_samples_split=int(min_samples_split),
                        max_features=min(max_features, 0.999),
                        max_depth=int(max_depth),
                        random_state=90,
                        n_jobs=-1),
            X_train_rf, y_train_rf, scoring='f1', cv=5
        ).mean()
    return val

pbounds = {'n_estimators': (50, 250), 'min_samples_split': (2, 25),'max_features': (0.
1, 0.999),'max_depth': (5, 12)}

RF_bo = BayesianOptimization(f=RF_evaluate, pbounds=pbounds, verbose=2,
random_state=1,)

RF_bo.maximize(init_points=5,n_iter=10,acq='ei')
print(RF_bo.max)
```

iter	target	max_depth	max_fe...	min_sa...	n_esti...
1	0.6972	7.919	0.7476	2.003	110.5
2	0.7024	6.027	0.183	6.284	119.1
3	0.6958	7.777	0.5844	11.64	187.0
4	0.6917	6.431	0.8894	2.63	184.1
5	0.6954	7.921	0.6023	5.229	89.62
6	0.6892	5.755	0.6416	6.509	119.8
7	0.701	8.098	0.4717	14.6	198.8
8	0.7121	9.364	0.3007	24.06	210.9
9	0.6892	5.148	0.8263	24.19	190.4
10	0.7031	7.551	0.1201	5.506	244.3
11	0.6997	6.354	0.1175	6.095	119.2
12	0.7057	8.763	0.3337	23.21	211.3
13	0.7049	8.615	0.9072	23.93	211.1
14	0.7107	9.447	0.619	23.97	211.8
15	0.7191	10.36	0.931	23.36	210.8

Les Paramètres sélectionnés : max_depth=10, max_features=0.93, min_samples_split=23, n_estimators=211

Modèle : Random Forest Classifier

Le résultat du classificateur (Utiliser les paramètres sélectionnés par GridSearchCV)

	precision	recall	f1-score	support
0	0.88	0.66	0.75	95547
1	0.31	0.63	0.42	23262
accuracy			0.66	118809
macro avg	0.60	0.65	0.59	118809
weighted avg	0.77	0.66	0.69	118809

Le résultat du classificateur (Utiliser les paramètres sélectionnés par BayesianOptimization)

	precision	recall	f1-score	support
0	0.86	0.77	0.81	95547
1	0.33	0.47	0.39	23262
accuracy			0.71	118809
macro avg	0.59	0.62	0.60	118809
weighted avg	0.75	0.71	0.73	118809

Résultat et discussion

Model	Score
RandomForest	0.78
LogisticReg	0.66

Ce qu’on a bien fait :

sélection des features, traitement du déséquilibre de l'échantillon, différents classificateurs sont utilisés.

Ce qu’on a mal fait :

mauvaise optimisation du modèle et faibles résultats de prédiction

Contributions

- **Idée du projet :** WANG
- **Code :**
  - Partie “Traitement des données（Numerical Features）”: essentiellement pompée sur Kaggle, adaptée par TONG.
  - Partie “Traitement des données（Categorical Features）”: WANG
  - Partie “Feature engineering”: Surtout TONG
- **Nettoyage des données**: Les 2, mais surtout TONG
- **Tuning des paramètres:** Surtout TONG
- **Algo Logistic Regression :** TONG
- **Algo Random Forest :** WANG
- **Slides :** Les 2, mais surtout WANG