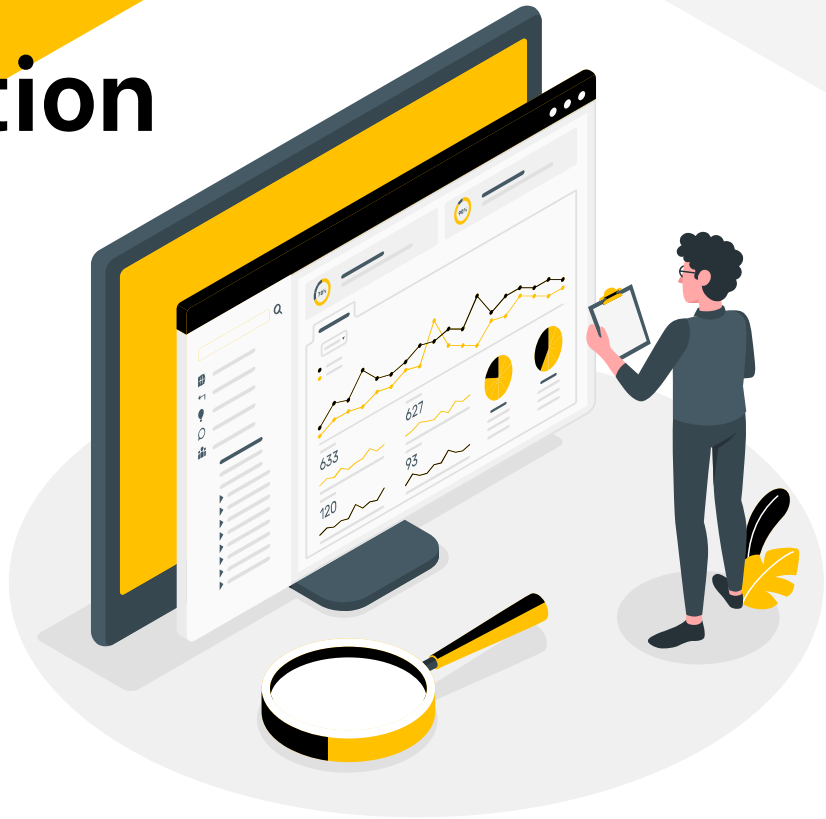
IA pour la prédiction des faillites d'entreprises



Adil / Véronique

Sommaire

Le dataset

01

- 6819 lignes et 96 colonnes
- 5 features (toutes type float sauf 'Net income flag' et 'Liability-Assets Flag' qui sont des int)

```
0    6811
1       8
Name: Liability-Assets Flag, dtype: int64
1    6819
Name: Net Income Flag, dtype: int64
```

- 1 target : Bankrupt? (valeur 0 si l'entreprise n'a pas fait faillite et valeur 1 si l'entreprise a fait faillite)

Exploration

02

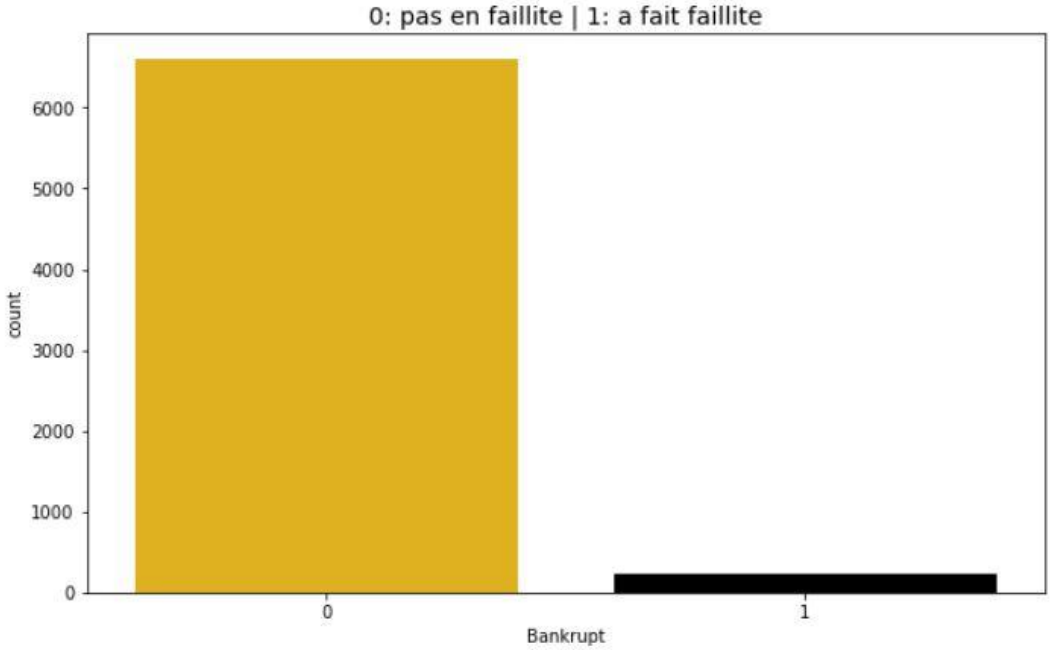
Notre cible : Bankrupt

0 = non faillite

6599 entreprises

1 = faillite

220 entreprises



Notre cible en %

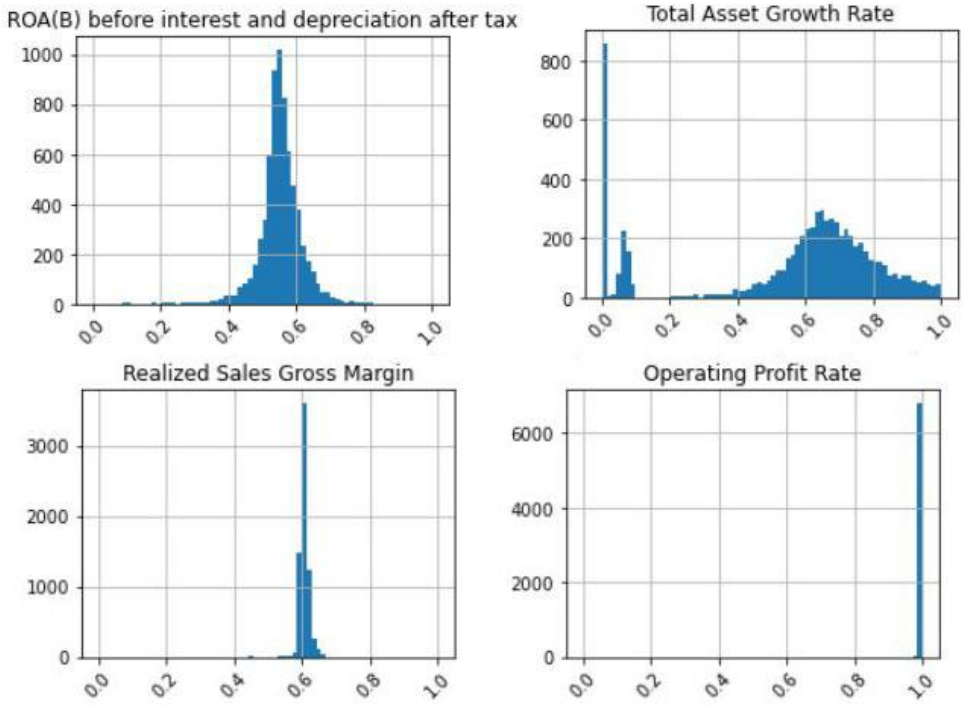
3% d'entreprises en faillites

97% d'entreprises qui n'ont pas fait faillite

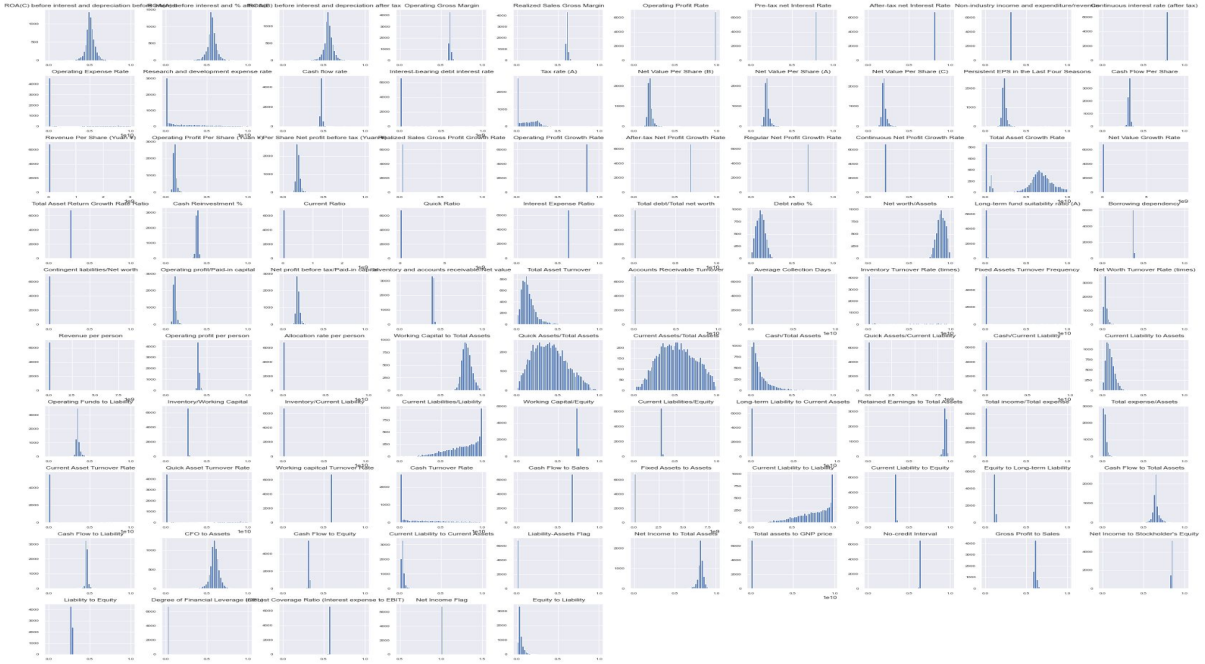
Les features

Nos 95 features ont des distributions et échelles différentes.

- 69 features ont des données entre 0 et 1
- 1 feature a juste des 1
- 1 feature a des 0 et 1
- les autres ont des échelles de 0 jusqu'à 9.990000e+09



Distribution des données



Feature correlation

	feature_1	feature_2	correlation
	Bankrupt?	Debt ratio %	0.250161
	Bankrupt?	Current Liability to Assets	0.194494
	Bankrupt?	Borrowing dependency	0.176543
	Bankrupt?	Current Liability to Current Assets	0.171306
	Bankrupt?	Liability to Equity	0.166812

	Bankrupt?	ROA(C) before interest and depreciation...	-0.260807
	Bankrupt?	ROA(B) before interest and depreciation...	-0.273051
	Bankrupt?	ROA(A) before interest and % after tax	-0.282941
	Bankrupt?	Net Income to Total Assets	-0.315457

3. Nos statégies

SCALING

Standardiser et normaliser les données

SMOTE

Oversampler les données

Preprocessing workflow

Preprocessing pipelines

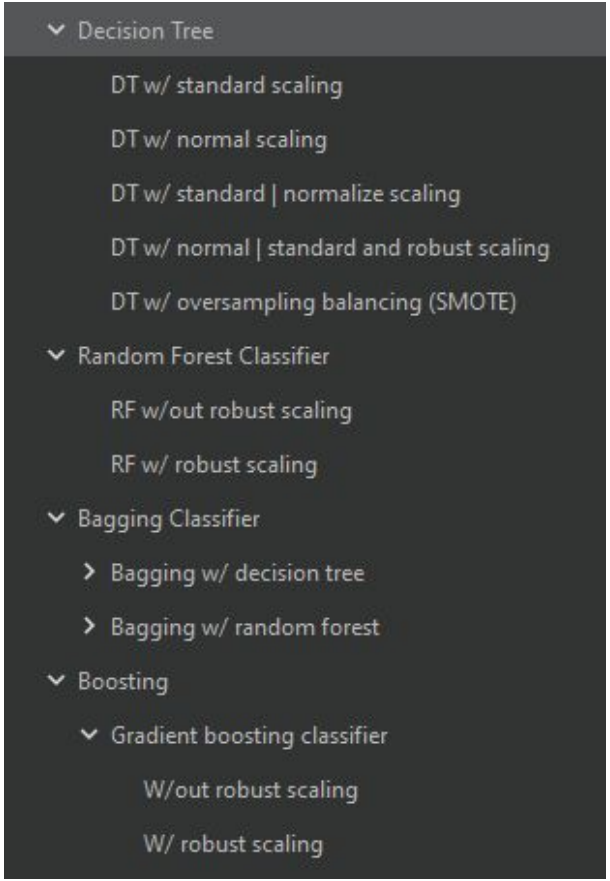
```
# checking features which do or do not have a normal distribution
def check_not_normal(X):
    data = []
    for i in X.columns:
        if X[i].skew() < -0.9 or X[i].skew() > 0.9:
            data.append(i)

    return data

def check_normal(X):
    data = []
    for i in X.columns:
        if X[i].skew() > -0.9 and X[i].skew() < 0.9:
            data.append(i)

    return data

ftr_to_scale = check_not_normal(X)
ftr_to_norm = check_normal(X)
```

```
scaling_itr = ColumnTransformer([
    ('scaling', StandardScaler(), ftr_to_scale)
], remainder='passthrough')
```

```
normalize_itr = ColumnTransformer([
    ('normal scaling', MinMaxScaler(), ftr_to_norm),
], remainder='passthrough')
```

```
scaling_nrm_itr = ColumnTransformer([
    ('normal scaling', MinMaxScaler(), ftr_to_norm),
    ('standard scaling', StandardScaler(), ftr_to_scale)
], remainder='passthrough')
```

```
robust_sc_itr = Pipeline([
    ('robust scaling', RobustScaler())
])
```



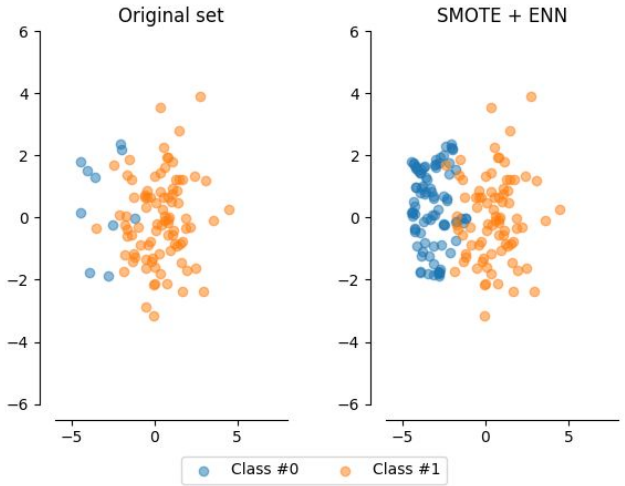
Preprocessing workflow

Résultats

1. RobustScaler sur toutes les features

2. SMOTE ENN : SMOTE est appliqué pour créer des données synthétiques d'échantillons de classes minoritaires, puis en utilisant ENN, les points de données sur la frontière sont supprimés pour augmenter la séparation des deux classes.

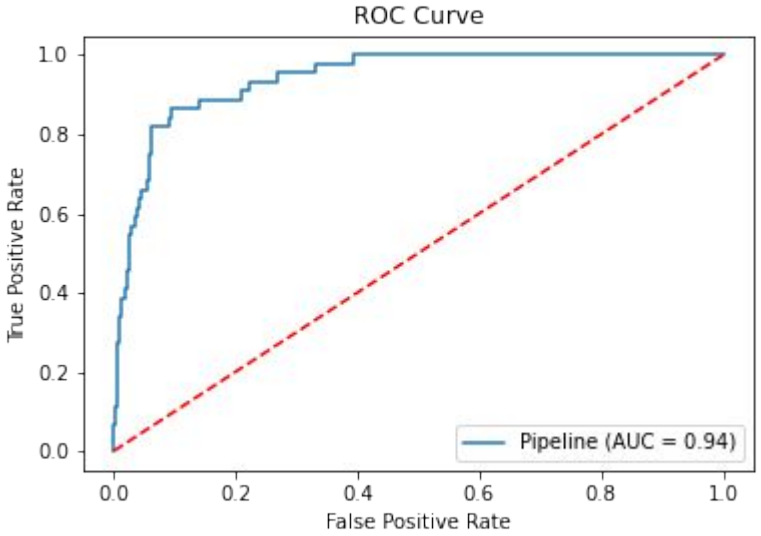
3. Random Forest
(avec GridSearchCV)
'max_depth': 10,
'n_estimators': 150



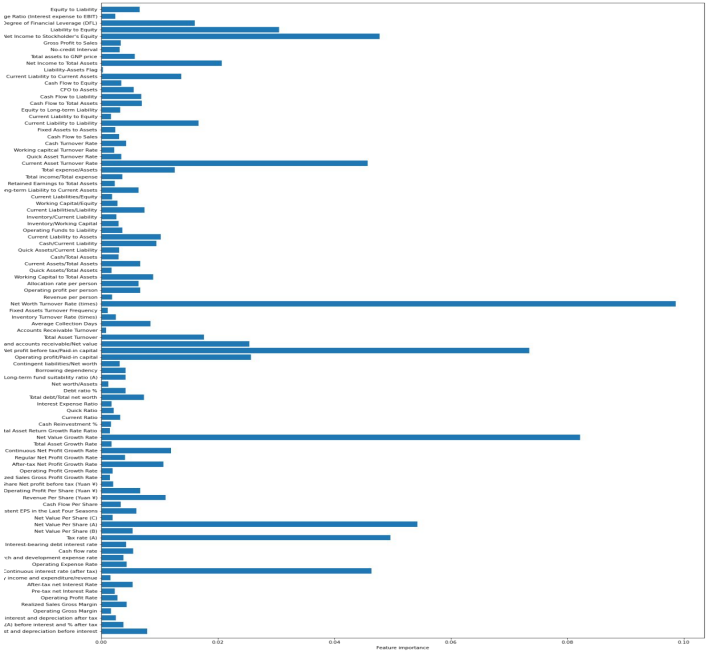
Recall score :

0 : 0.93
1 : 0.77

Metrics

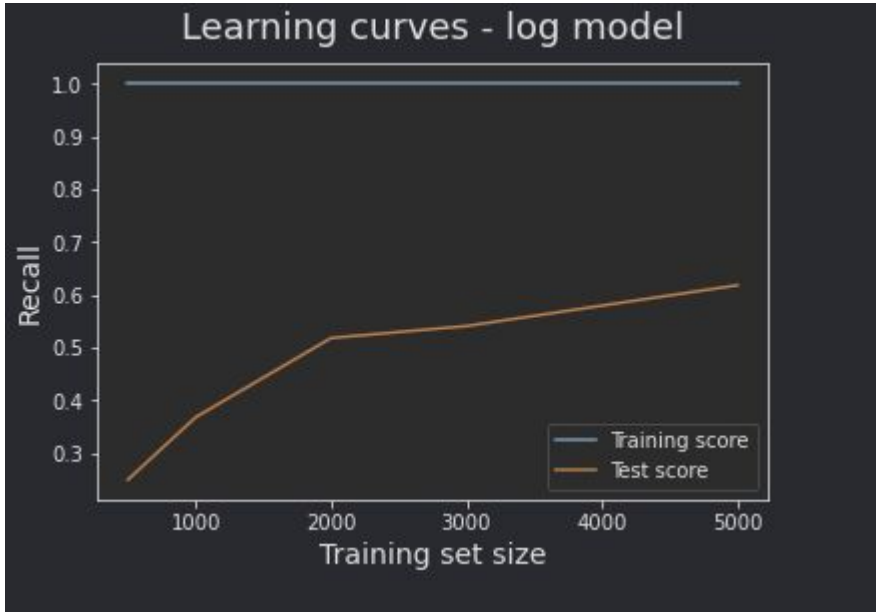


Feature



Net Worth Turnover Rate (Times)
Net Value Growth Rate

Generalisation



Conclusion

Cette prédiction n'est pas assez précise pour être commercialisé à cause de sa marge d'erreur encore trop importante.

A voir par la suite :

- Features selection
- Model tuning
- Autres modèles