Prédiction de faillite des entreprises

Rendu de projet Exploitation des données massives en finance

DURIEZ Typhanie, ZEMRAK Ianis

A partir d'une base de données listant des éléments de bilan d'entreprises, l'objectif est de classer les entreprises en deux groupes distincts : les entreprises ayant fait faillite et celles dont ce n'est pas le cas. La base de données regroupe 95 variables du bilan de ces entreprises (ROA, Operating Profit…) ainsi qu'une variable binaire (Bankrupt?) prenant la valeur 1 si l'entreprise est en défaut, 0 sinon.

Compte tenu des données à disposition ainsi que du problème à traiter, nous sommes dans un cas de classification supervisée. Ainsi nous avons adapté notre démarche en conséquence.

I – Observation des données

La base de données transformée en dataframe « df » regroupe 6819 entreprises parmi lesquelles 220 sont considérées en faillite. Nous avons en premier lieu observé la corrélation des variables avec la variable « Bankrupt? » en isolant les variables ayant une corrélation supérieure à 0,25 avec cette dernière. Ces 6 variables isolées dans le dataframe « df_highest_corr » n'ont pas une corrélation suffisamment importante avec le risque de défaut pour qu'on puisse baser notre étude sur ces éléments, la corrélation la plus élevée étant seulement de 0,315.

A ce stade nous avons cherché à éliminer certaines variables par leurs variances mais étant trop faibles pour l'ensemble des variables elle est inexploitable.

II – Normalisation

Les variables étant déjà au niveau il ne nous était pas nécessaire de les normaliser, cependant nous avons quand même inséré un code de normalisation au cas où nous souhaiterions utiliser ce code sur des données nécessitant d'être normalisées.

III – Training et sélection du modèle

Après avoir défini X, y et séparé les données nous appliquons trois modèles pour voir comment ils se comportent face à notre jeu de données.

Linear Discriminant Analysis :

On applique l'analyse discriminante linéaire sur nos données avec X_train (auquel on a retiré la variable « Bankrupt ? ») et y_train[« Bankrupt ? »] que l'on cherche à expliquer. Le modèle une fois entrainé est appliqué à X_test et on obtient ainsi y_pred_lda que l'on compare à y_test via la matrice de confusion et le f1 score ci-dessous.

```
In [288]: lda_confusion_matrix = confusion_matrix(y_test, y_pred_lda)
          lda_confusion_matrix #permet de voir la répartition dans les classes

Out[288]: array([[1292,   13],
                 [  44,   15]], dtype=int64)

In [289]: print("Score :", f1_score(y_test, y_pred_lda, average='micro'))

          Score : 0.9582111436950147
```

Comme observé le modèle se trompe sur 13 faux positifs et 44 faux négatifs. On remarque une tendance à classer davantage dans la catégorie non-défaut surement dû au fait que près de 97% de la base des données sont des entreprises n'ayant pas fait faillite.

Cependant ces erreurs sont très raisonnables et le score du modèle est d'ailleurs bon.

Support Vector Machine :

Comme pour le LDA on applique le SVM sur nos données avec X_train (auquel on a retiré la variable « Bankrupt ? ») et y_train[« Bankrupt ? »] que l'on cherche à expliquer. Le modèle une fois entrainé est appliqué à X_test et on obtient ainsi y_pred_svm que l'on compare à y_test via la matrice de confusion et le f1 score ci-dessous.

```
In [296]: svm_confusion_matrix = confusion_matrix(y_test, y_pred_svm)
          svm_confusion_matrix

Out[296]: array([[1305,    0],
                 [  59,    0]], dtype=int64)

In [297]: print("Score :", f1_score(y_test, y_pred_svm, average='micro'))

          Score : 0.9567448680351907
```

Le score parait bon mais est inexploitable. En effet comme on peut le voir dans la matrice de confusion, le SVM classe tous les individus comme ne faisant pas faillite (on retrouve donc 59 faux positifs et 0 faux négatifs). En faisant cela, puisque 97% des entreprises ne font pas faillite il obtient un très bon score de 0,9567 soit environ la même proportion que notre base de données.

Le SVM est donc inutilisable malgré son score très bon.

Random Forest :

De la même manière on soumet les données avec y_train que l'on cherche à expliquer au Random Forest Classifier. On obtient y_pred_rfc que l'on évalue via la matrice de confusion et le f1 score suivants :

```
In [47]: rfc_confusion_matrix = confusion_matrix(y_test, y_pred_rfc)
         rfc_confusion_matrix

Out[47]: array([[1318,    6],
                [  31,    9]], dtype=int64)

In [48]: print("Score :", f1_score(y_test, y_pred_rfc, average='micro'))

         Score : 0.9728739002932552
```

Le score du modèle est bon (0,97) et il y a bien des faux négatifs ce qui nous permet d'utiliser ce modèle car il semble adapté pour la suite de notre analyse.

IV – Sélection des variables

Grace à la fonctionnalité rfc.feature_importances_ on obtient l'importance de chacune des variable dans le Random Forest.

Nous sélectionnons les 8 variables les plus importantes du modèle (voir annexe 1) et recréons un nouveau modèle entrainé avec seulement ces 8 variables pour voir si son score diminue drastiquement par rapport au modèle initial (donc en ayant retiré 87 variables). Le score obtenu est le suivant :

```
In [151]: rfc_confusion_matrix = confusion_matrix(y_test, y_pred_rfc)
          rfc_confusion_matrix

Out[151]: array([[1303,   14],
                 [  33,   14]], dtype=int64)

In [152]: print("Score :", f1_score(y_test, y_pred_rfc, average='micro'))

          Score : 0.9655425219941349
```

Le score est très bon (0,96) et la classification est toujours bien répartie entre positifs et négatifs.

Nous poursuivons davantage en retirant de nouvelles variables et en gardons seulement 3. Nous entrainons un nouveau modèle et obtenons les résultats suivants :

```
In [158]: rfc_confusion_matrix = confusion_matrix(y_test, y_pred_rfc)
          rfc_confusion_matrix

Out[158]: array([[1298,    2],
                 [  57,    7]], dtype=int64)

In [159]: print("Score :", f1_score(y_test, y_pred_rfc, average='micro'))

          Score : 0.9567448680351907
```

Le score est bon mais on remarque une forte répartition du côté des faux négatifs.

Continuons en ne conservant qu'une seule variable et en réentraînant un nouveau modèle.

```
In [166]: rfc_confusion_matrix = confusion_matrix(y_test, y_pred_rfc)
          rfc_confusion_matrix

Out[166]: array([[1296,   30],
                 [  32,    6]], dtype=int64)

In [167]: print("Score :", f1_score(y_test, y_pred_rfc, average='micro'))

          Score : 0.9545454545454546
```

Le score reste excessivement élevé (0,95) avec seulement une variable pour classer. La répartition est également très bien répartie puisqu'assez équilibrée entre faux négatifs (32) et faux positifs (30).

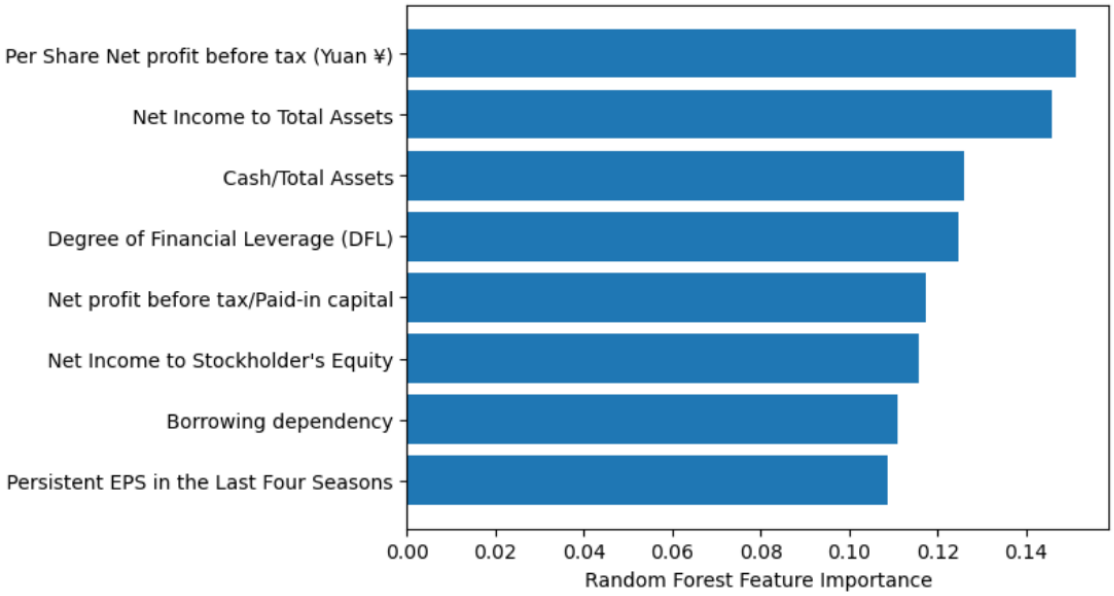
V - Interprétation des résultats et commentaire

La première conclusion hâtive serait de dire qu'une seule variable bien sélectionnée (la plus importante) suffit à prédire la faillite ou non d'une entreprise puisque le score est excellent (0,95) pour le modèle avec une seule variable et la répartition faux positifs-faux négatifs semble équilibrée.

Cependant au vu de tous les modèles entrainés et les différentes méthodes (LDA, SVM, RFC) un élément saute aux yeux : peu importe la répartition dans les matrices de confusion, les f1 scores sont toujours supérieurs à 0,95 ce qui est anormal. De plus, il serait étrange qu'avec une seule variable le Random Forest puisse prédire à 97% les entreprises qui vont faire faillite.

L'explication qui nous vient est que compte tenu de la prédominance des entreprises ne faisant pas faillite (97%) dans le jeu de données, peu importe la répartition que le modèle effectue, du moment que la très grande majorité est classée en non-faillite il aura toujours un score très élevé.

Une amélioration possible serait d'augmenter la part des entreprises faisant faillite pour empêcher le modèle de pouvoir faire une répartition aléatoire et lui donner plus de matière pour cerner les entreprises faisant faillite. La base de données ne contient pas assez d'entreprises en faillite pour pouvoir se fier aux scores obtenus.



Annexe 1 : Variables les plus importantes d'après le modèle Random Forest