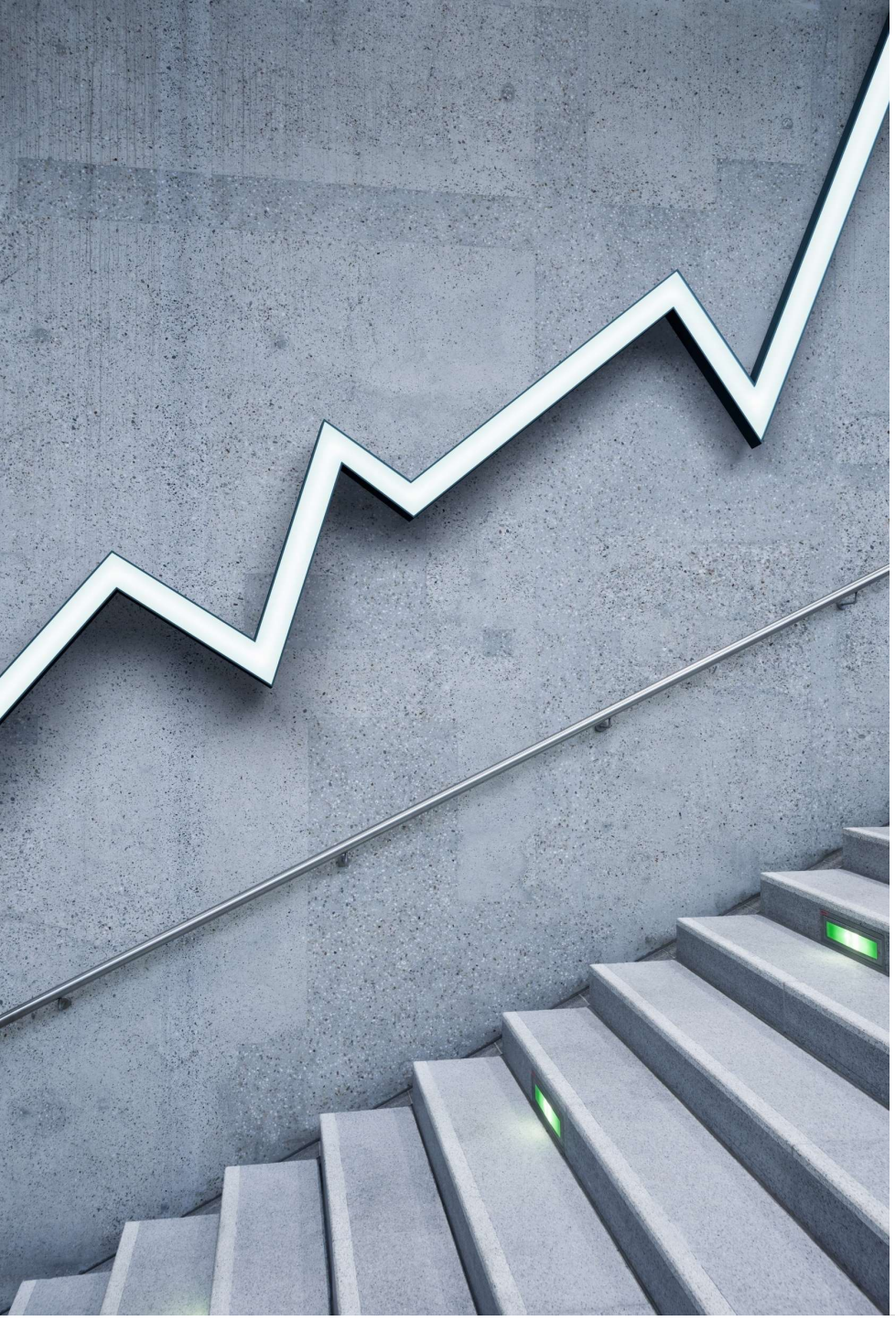


22/03/2022

Soutenance PROJET 7

IMPLÉMENTEZ UN MODÈLE DE SCORING

ADAMA TOURE

FORMATION DATA SCIENTIST OPENCLASSROOMS

SOMMAIRE

I – Introduction générale du sujet

La société financière « PRET A DEPENSER » souhaite mettre en place un outil de “scoring crédit” pour calculer la probabilité qu’un client rembourse son crédit. Elle mettra à disposition de ses chargés clients un dashboard transparent interactif pour les aider à la prise de décisions et surtout à les justifier auprès des clients.

Pour cela, nous disposons de :

- sources de données variées (données comportementales, données provenant d'autres institutions financières, etc...).

- un kernel Kaggle pour faciliter la préparation des données nécessaires à l’élaboration du modèle de scoring

MISSION :
Construire un modèle de scoring qui donnera une prédiction sur la probabilité de faillite d'un client de façon automatique et construire ce dashboard

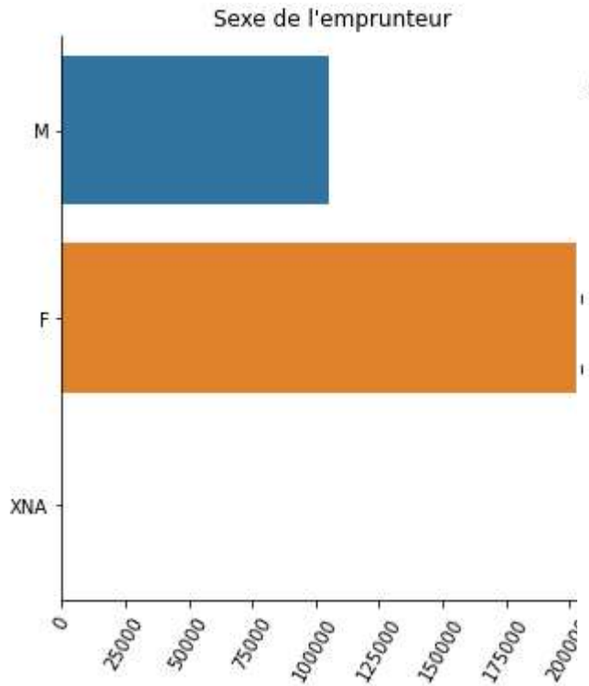
II – Analyse exploratoire des données

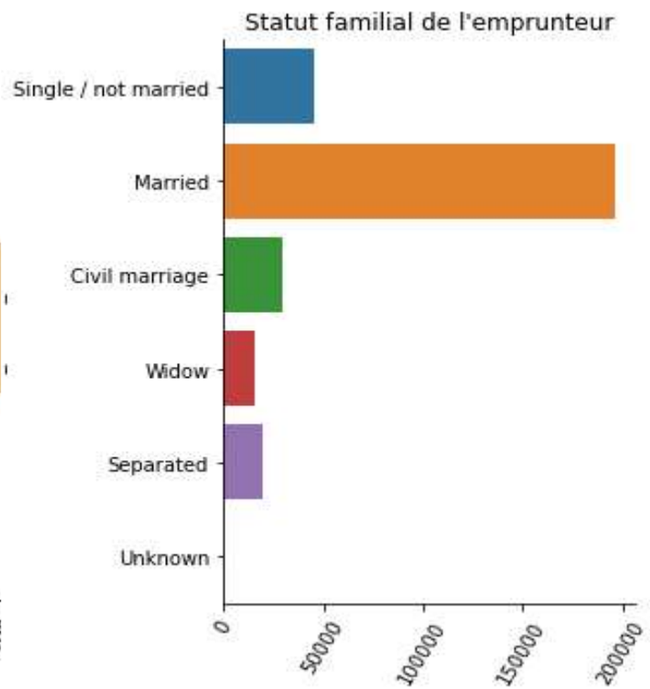
- Un échantillon de 8 bases de données (fichiers csv) contenant
  - des informations sur plus de 300 K individus labellisés
  - environ 50 K non labellisés.

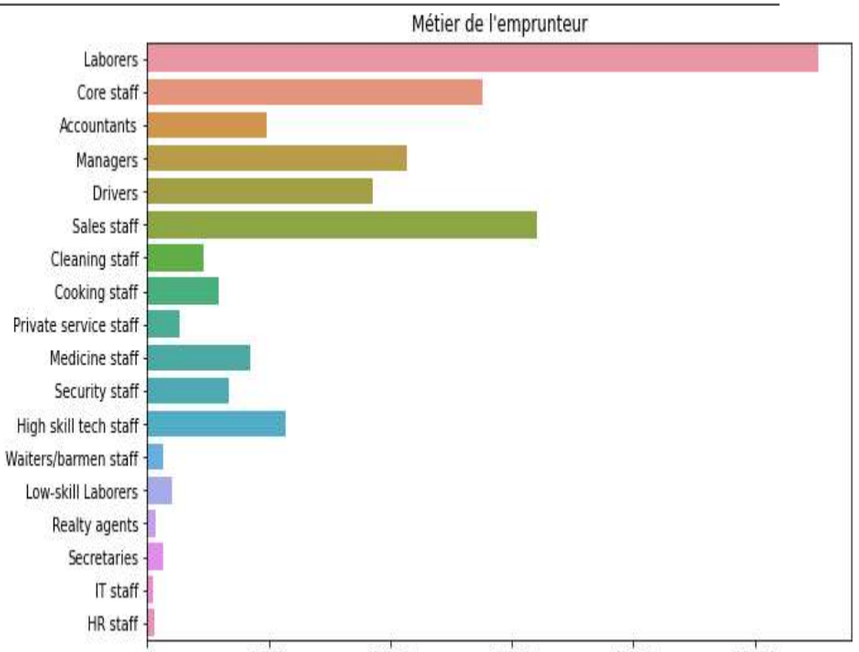
- Un nombre des variables ont une quantité significative de valeurs manquantes et aberrantes.
  - - Agrégation de tableau
  - - Remplissage en fonction de la typologie de variable (Mode vs Moyenne)

- Un jeu de données est très déséquilibré avec une proportion 95 % - 5 % dans la variable cible.
  - - Rééquilibrage des données par SMOTE

Quelques analyses



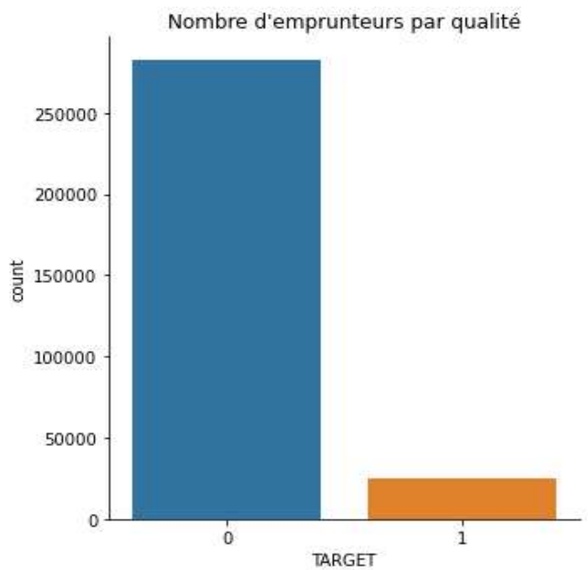




III – Entrainements de modèles

Définition de la baseline modèle

- Déséquilibre 95 % - 5 %



Utilisation SMOTE 50 % - 50 %

Tests de modèles : Logistic Regression et kNN

```
# Model 1: train the model on the training set
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred_knn))
```

```
0.922
```

```
# Model 2: train the model on the training set
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# make predictions on the testing set
y_pred_LR = logreg.predict(X_test)

# compare actual response values (y_test) with predicted response values (y_pred)
print(metrics.accuracy_score(y_test, y_pred_LR))
```

```
0.9055
```

Meilleurs que la baseline modèle

Hyper Optimisation de ces modèles : GridSearchCV

```
#In case of classifier like knn the parameter to be tuned is n_neighbors
start = time()
param_grid = {'n_neighbors':np.arange(2,15)}
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(X,y)
```

```
#Logistic Regression requires two parameters 'C' and 'penalty' to be optimised
start = time()
grid={"C":np.logspace(-3,3,7), "penalty":["l2","none"]}
logreg_cv=GridSearchCV(logreg,grid,cv=5)
logreg_cv.fit(X,y)
```

Meilleurs que la baseline modèle

Hyper Optimisation de ces modèles : GridSearchCV

```
# Model 1 optimized
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
y_pred_knn_opt = knn.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred_knn_opt))
```

```
0.9082204450247539
```

```
print(confusion_matrix(y_test, y_pred_knn_opt))
print(metrics.accuracy_score(y_test, y_pred_knn_opt))
print(classification_report(y_test, y_pred_knn_opt))
```

```
[[15727  2609]
 [  765 17661]]
0.9082204450247539
              precision    recall  f1-score   support

         0.0       0.95      0.86      0.90     18336
         1.0       0.87      0.96      0.91     18426

    accuracy                           0.91     36762
   macro avg       0.91      0.91      0.91     36762
weighted avg       0.91      0.91      0.91     36762
```

```
# Model 2 optimized
logreg = LogisticRegression(class_weight='balanced', C= 0.001, penalty='l2', max_iter=1000)
logreg.fit(X_train, y_train)

# make predictions on the testing set
y_pred_LR_opt = logreg.predict(X_test)

# compare actual response values (y_test) with predicted response values (y_pred)
print(metrics.accuracy_score(y_test, y_pred_LR_opt))
```

```
0.9544715447154472
```

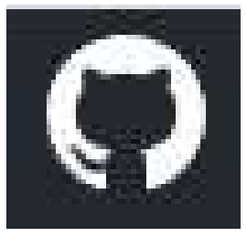
```
print(confusion_matrix(y_test, y_pred_LR_opt))
print(metrics.accuracy_score(y_test, y_pred_LR_opt))
print(classification_report(y_test, y_pred_LR_opt))
```

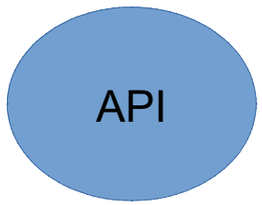
```
[[1794   21]
 [ 147 1728]]
0.9544715447154472
              precision    recall  f1-score   support

         0.0       0.92      0.99      0.96      1815
         1.0       0.99      0.92      0.95      1875

    accuracy                           0.95      3690
   macro avg       0.96      0.96      0.95      3690
weighted avg       0.96      0.95      0.95      3690
```

IV – Création d’API et déploiement sur le web

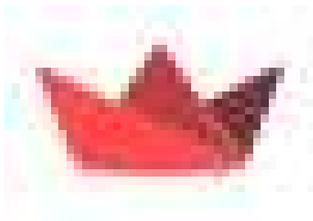
Outils utilisés :

- Git 

- Github 



- Heroku

- Streamlit 

V – Implémentation de dashboard

Démonstration

VI – Conclusion et recommandations

La société dispose d'un modèle de scoring des clients utilisant la Logistic Regression

Un dashboard utilisable par ses conseillers – clients avec un niveau de transparence demanadées par les clients.

Toutefois, en recommandation, il serait pertinent que la société constitue ses propres indicateurs/données en interne pour être de moins en moins dépendante des sources de données externes avec les risques inhérents (validité des données, mises à jour, restrictions d'accès, conformités réglementaires.

Vérifier régulièrement la validité du modèle de classification trouvé à ce moment de l'étude.

MERCI