Modélisation du risque de défaut avec les algorithmes de machine learning

Réalisé par :

Assedra Nassim

Barzouq khadija

Nawfal el malki

Résume :

La dernière crise financière a mis en exergue la nécessité pour les sociétés de crédit de se doter de modèles dynamiques et précis d'évaluation du risque de défaut de leurs clients. A cet effet la quantification de la probabilité de défaut et sa prédiction constitue, un jalon incontournable des activités de gestion du risque. Dans ce cadre, le présent projet propose des modèles statistiques qui ont pour but la prédiction du défaut et la détermination des variables qui contribuent à la survenance évènement. Ceci, tout en mettant en relief la pertinence de ces modèles à travers leur propension à prédire et à capter la dynamique du défaut et en apportant des évidences empiriques et des analyses

1. Introduction :

La modélisation du risque de crédit constitue un véritable défi. En effet, une des difficultés majeures des directions de risque de crédit, réside dans l'évaluation de la probabilité de défaut individuelle de leurs clients. L'hypothèse d'asymétrie d'information est centrale dans les modèles, car on suppose que l'emprunteur détient plus d'information sur sa probabilité de défaut. A cet effet, les institutions de crédit utilisent différentes méthodes pour se prémunir de cette asymétrie informationnelle. On peut citer : l'auto-sélection des risques, les collatéraux ou les garanties de prêts et le scoring bancaire. Diverses variables tant d'ordre qualitatifs que quantitatifs, sont utilisées pour spécifier les probabilités de défaut des emprunteurs.

L'objectif principal de ce projet consiste donc à l'analyse des différents risques des prêts octroyés, en estimant des modèles statistiques en vue de prédire la défaillance des clients. Pour ce faire, certaines données disponibles sur le site Kaggle seront utilisées afin de déterminer quelles sont les variables significatives dans la prédiction du défaut. De plus, on s'appuiera sur des méthodes de classification dont le but est principalement d'identifier la classe au quelle appartient un client potentiel.

2. Définition du risque de crédit

Pour bien comprendre le risque de crédit, il est nécessaire de revenir dans un premier temps sur une distinction conceptuelle ainsi qu'à la source de ce risque. Le défaut survient lorsque le client ne peut plus faire face à ses engagements vis à vis de ses créanciers.

En terme probabiliste le risque de crédit représente la probabilité de défaut de paiement d'un client, il ne se limite pas uniquement à ce risque de défaillance, mais aussi à d'autres risques qui peuvent compromettre la rentabilité de l'établissement financier. Les plus importants sont le risque de liquidité, le risque de transaction et le risque de marché. Nous allons ici, mettre en avant le risque de crédit au détriment des autres.

La probabilité de défaut d'un client est le risque que sa note baisse pendant la période à venir. Plus sa note initiale est bonne, moins cette probabilité est importante.

3. Cadre pratique du projet

Dans cette partie nous allons procéder à la préparation des données ainsi que la modélisation du risque de défaut. En effet le prétraitement des données est une étape cruciale dans toute analyse en vue de vérifier avec soin toutes les variables et de déceler d'éventuelles erreurs. L'objectif est d'éliminer les données de mauvaise qualité (redondantes, incomplètes ou incorrectes) et de commencer à créer les données qui peuvent garantir un environnement sain pour mener une modélisation tangible.

Nous formerons ensuite aux méthodes d'apprentissage automatique dans le contexte spécifique de la modélisation du risque de crédit. Nous mettrons en exergue les commandes de mise en œuvre des méthodes évoquées précédemment puis nous étudierons leurs per□formances.

4. Présentation du cadre du travail

Dans le présent travail, les modèles en étude sont des modèles à variables dépendantes dichotomiques. Nous devons construire un modèle en mesure de prédire la probabilité de défaut des clients. On est alors amené à réaliser des modèles statistiques qui vont permettre la prévention du risque crédit inhérent à son activité.

5. Base de données

Pour réaliser notre projet on a fait une extraction des données à partir du site kaggle. On a utilisé un code pour le scrapping de la base. On a rencontré une difficulté au niveau du scrapping car la base qu'on a choisie avait la main directement pour le téléchargement. On a reussis apres pas mal de tentative de faire un code pour le scrapping qui extracte la base de données et l'enregistre directement dans un chemin donné.

6. Compréhension des variables

La base de données sujette de notre étude contient des informations sur 500 clients relatives à 11 variables. Il est important de comprendre la signification de chacune de ces variables.

loan_status : qui est la variable à expliquer, et qui représente soit la défaillance ou non du client.

person_age : L'âge du client

person_income : représente le revenu du client

person_home_ownership : combien de bien immobilier il possède

person_emp_length : l'ancienneté au travail du client

loan_intent : l'intention pour laquelle le prêt est demandé

loan_grade : Classification du crédit

loan_amnt : le montant du crédit

cb_person_cred_hist_length : la durée de credit

loan_int_rate : le taux d'interet

7. Traitement de la base de données :

1. Les valeurs manquantes :

En affichant les valeurs manquantes (Voir le fichier du script), On a remarqué qu'on a des données manquantes au niveau de la variable "person_emp_length" qui réfère aux nombres d'années d'emplois avec un pourcentage de 3% à peu près ainsi que 9.5% des données de la variable "loan_int_rate".

Puisque on a une grande base de données et que le pourcentage des données manquants au niveau de la variable "person_emp_length" on va supprimer ces données manquantes.

Pour la variable qui reste on va faire une imputation avec la méthode du plus proche voisin.

2. Les valeurs aberrantes :

Maintenant qu'on n'a plus de valeurs manquantes, on va passer au traitement des valeurs aberantes. Toutes les valeurs des variables "person_emp_length" (représentant la durée d'emploi) qui sont supérieures à person_age (âge) sont des valeurs aberrantes car elles sont irréelles. Ici, on a donc 2 valeurs aberrantes pour 2 catégories d'âge : 21 ans et 22 ans. On va donc les remplacer respectivement par la moyenne des durées d'emploi des personnes ayant 21 et 22 ans.

A partir de plus de 40 ans de carrière, il y a plus de probabilité de retrouver des valeurs aberrantes. Ici, la personne en question a 78 ans, donc il est possible que cette personne ait bien 40 ans de carrière. Maintenant que notre base de données est clean, on pourra commencer la construction de notre modèle. Mais avant cela, on va vérifier s'il existe un déséquilibre ou non de notre base de données.

8. Aperçu sur les modèles utilisés

1. la régression logistique

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y. Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien.

Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

Mathématiquement, on considère une entrée $X= x1\ x2\ x3\ \dots\ xn$, la régression logistique a pour objectif de trouver une fonction h telle que nous puissions calculer :

$$y= \{1 \quad si \quad hX \geq seuil\ ,\ 0 \quad si \quad hX < seuil\}$$

On comprend donc qu'on attend de notre fonction h qu'elle soit une probabilité comprise entre 0 et 1, paramétrée par =1 2 3 n à optimiser, et que le seuil que nous définissons correspond à notre critère de classification, généralement il est pris comme valant 0.5.

La fonction qui remplit le mieux ces conditions est la fonction sigmoïde, définie sur R à valeurs dans [0,1]. Elle s'écrit de la manière suivante :

Graphiquement, celle-ci correspond à une courbe en forme de S qui a pour limites 0 et 1 lorsque x tend respectivement vers $-\infty$ et $+\infty$ passant par $y = 0.5$ en $x = 0$.

2. Random forest

Une Random Forest (ou Forêt d'arbres de décision en français) est une technique de Machine Learning très populaire auprès des Data Scientists et pour cause : elle présente de nombreux avantages comparé aux autres algorithmes de data.

C'est une technique facile à interpréter, stable, qui présente en général de bonnes accuracies et qui peut être utilisée pour des tâches de régression ou de classification. Elle couvre donc une grande partie des problèmes de Machine Learning.

Dans Random Forest il y a d'abord le mot « Forest » (ou forêt en français). On comprend donc que cet algorithme va reposer sur des arbres que l'on appelle arbre de décision ou arbre décisionnel.

Random Forest est ce qu'on appelle une méthode d'ensemble (ou ensemble method en anglais) c'est-à-dire qu'elle « met ensemble » ou combine des résultats pour obtenir un super résultat final.

Mais les résultats de quoi ? Tout simplement des différents arbres de décision qui la composent.

Les Random Forest peuvent être composées de plusieurs dizaines voire centaines d'arbres, le nombre d'arbre est un paramètre que l'on ajuste généralement par validation croisée (ou cross-validation en anglais). Pour faire court, la validation croisée est une technique d'évaluation d'un algorithme de Machine Learning consistant à entrainer et tester le modèle sur des morceaux du dataset de départ.

Chaque arbre est entraîné sur un sous-ensemble du dataset et donne un résultat (oui ou non dans le cas de notre exemple sur les champignons). Les résultats de tous les arbres de décision sont alors combinés pour donner une réponse finale. Chaque arbre « vote » (oui ou non) et la réponse finale est celle qui a eu la majorité de vote.

3. Méthode de bagging

On découpe notre dataset en plusieurs sous-ensembles aléatoirement constitués d'échantillons – d'où le « Random » dans Random Forest.

On entraine un modèle sur chaque sous-ensemble : il y a autant de modèles que de sous-ensembles.

On combine tous les résultats des modèles (avec un système de vote par exemple) ce qui nous donne un résultat final.

De cette manière on construit un modèle robuste à partir de plusieurs modèles qui sont pas forcément aussi robustes.

En bref, cet algorithme est très populaire pour sa capacité à combiner les résultats de ses arbres pour obtenir un résultat final plus fiable. Son efficacité lui a permis d'être utilisé dans de nombreux domaines comme par exemple le marketing téléphonique pour prédire le comportement de clients ou encore la finance pour la gestion de risques.

4. les machines à vecteurs de support

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais support-vector machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination note 1 et de régression. Les SVM sont une généralisation des classifieurs linéaires.

Les séparateurs à vaste marge ont été développés dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik sur le développement d'une théorie statistique de l'apprentissage : la théorie de Vapnik-Chervonenkis. Ils ont rapidement été adoptés pour leur capacité à travailler avec des données de grandes dimensions, le faible nombre d'hyper paramètres, leurs garanties théoriques, et leurs bons résultats en pratique.

Les SVM ont été appliqués à de très nombreux domaines (bio-informatique, recherche d'information, vision par ordinateur, finance1…). Selon les données, la performance des machines à vecteurs de support est de même ordre, ou même supérieure, à celle d'un réseau de neurones ou d'un modèle de mélanges gaussiens ;

5. Le boosting de gradient

Le Boosting de Gradient est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleure prédiction.

On parle d'ailleurs de méthode d'agrégation de modèles. L'idée est donc simple : au lieu d'utiliser un seul modèle, l'algorithme va en utiliser plusieurs qui serons ensuite combinés pour obtenir un seul résultat.

C'est avant tout une approche pragmatique qui permet donc de gérer des problèmes de régression comme de classification.

Pour décrire succinctement le principe, le l'algorithme travaille de manière séquentielle. Contrairement par exemple au Random Forest. cete façon de faire va le rendre plus lent bien sur mais il va surtout permettre à l'algorithme de s'améliorer par capitalisation par rapport aux exécutions précédentes. Il commence donc par construire un premier modèle qu'il va bien sur évaluer (on est bien sur de l'apprentissage supervisé). A partir de cette première évaluation, chaque individu va être alors pondérée en fonction de la performance de la prédiction. Etc.

XGBoost se comporte donc remarquablement dans les compétitions d'apprentissage automatique(Machine Learning), mais pas seulement grâce à son principe d'auto-amélioration séquentielle car il inclut en effet un grand nombre d'hyper paramètres qui peuvent être modifiés et réglés à des fins d'amélioration !

Cette grande flexibilité fait donc de XGBoost un choix solide qui vous faut absolument essayer.

Le Boosting de Gradient est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleure prédiction.

On parle d'ailleurs de méthode d'agrégation de modèles. L'idée est donc simple : au lieu d'utiliser un seul modèle, l'algorithme va en utiliser plusieurs qui serons ensuite combinés pour obtenir un seul résultat.

C'est avant tout une approche pragmatique qui permet donc de gérer des problèmes de régression comme de classification.

Pour décrire succinctement le principe, le l'algorithme travaille de manière séquentielle. Contrairement par exemple au Random Forest. Cette façon de faire va le rendre plus lent bien sur mais il va surtout permettre à l'algorithme de s'améliorer par capitalisation par rapport aux exécutions précédentes. Il commence donc par construire un premier modèle qu'il va bien sur évaluer (on est bien sur de l'apprentissage supervisé). A partir de cette première évaluation, chaque individu va être alors pondérée en fonction de la performance de la prédiction. Etc.

XGBoost se comporte donc remarquablement dans les compétitions d'apprentissage automatique(Machine Learning), mais pas seulement grâce à son principe d'auto-amélioration séquentielle …

XGBoost inclut en effet un grand nombre d'hyper paramètres qui peuvent être modifiés et réglés à des fins d'amélioration.

6. les k plus proches voisins

En intelligence artificielle, plus précisément en apprentissage automatique, la méthode des k plus proches voisins est une méthode d'apprentissage supervisé. En abrégé KPPV ou k-PPV en français, ou plus fréquemment k-NN ou KNN, de l'anglais k-nearest neighbors.

Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de N couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir. Puisque cet algorithme est basé sur la distance, la normalisation peut améliorer sa précision.1,2 Par exemple, dans un problème de classification, on retiendra la classe la plus représentée parmi les k sorties associées aux k entrées les plus proches de la nouvelle entrée x.

En reconnaissance de forme, l'algorithme des k plus proches voisins (k-NN) est une méthode non paramétrique utilisée pour la classification et la régression. Dans les deux cas, il s'agit de classer l'entrée dans la catégorie à laquelle appartient les k plus proches voisins dans l'espace des caractéristiques identifiées par apprentissage. Le résultat dépend si l'algorithme est utilisé à des fins de classification ou de régression :

en classification k-NN, le résultat est une classe d'appartenance. Un objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins, (k est un nombre entier positif généralement petit). Si k = 1, alors l'objet est affecté à la classe d'appartenance de son proche voisin.

en régression k-NN, le résultat est la valeur pour cet objet. Cette valeur est la moyenne des valeurs des k plus proches voisins.

La méthode k-NN est basée sur l'apprentissage préalable, ou l'apprentissage faible, où la fonction est évaluée localement, le calcul définitif étant effectué à l'issue de la classification. L'algorithme k-NN est parmi les plus simples des algorithmes de machines learning.

Que ce soit pour la classification ou la régression, une technique efficace peut être utilisée pour pondérer l'influence contributive des voisinages, ainsi les plus proches voisins contribuent-ils plus à la moyenne que les voisins plus éloignés. Pour exemple, un schéma courant de pondération consiste à donner à chaque voisin une pondération de 1/d, ou d est la distance de l'élément, à classer ou à pondérer, de ce voisin.

Les voisins sont pris depuis un ensemble d'objets pour lesquels la classe (en classification k-NN) ou la valeur (pour une régression k-NN) est connue. Ceci peut être considéré comme l'ensemble d'entraînement pour l'algorithme, bien qu'un entraînement explicite ne soit pas particulièrement requis.

Une particularité des algorithmes k-NN est d'être particulièrement sensible à la structure locale des données.

Les réseaux neuronaux, également appelés réseaux neuronaux artificiels ou réseaux neuronaux simulés, constituent un sous-ensemble de l'apprentissage automatique et sont au cœur des algorithmes d'apprentissage profond. Leur nom et leur structure sont inspirés du cerveau humain, imitant la façon dont les neurones biologiques se signalent les uns aux autres.

Les réseaux de neurones artificiels sont constitués de couches de nœuds, contenant une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque nœud, ou neurone artificiel, se connecte à un autre et possède un poids et un seuil associés. Si la sortie d'un nœud individuel est supérieure à la valeur seuil spécifiée, ce nœud est activé, envoyant des données à la couche suivante du réseau. Dans le cas contraire, aucune donnée n'est transmise à la couche suivante du réseau.

Les réseaux neuronaux s'appuient sur des données de formation pour apprendre et améliorer leur précision au fil du temps. Cependant, une fois que ces algorithmes d'apprentissage sont réglés avec précision, ils constituent des outils puissants en informatique et en intelligence artificielle, nous

permettant de classer et de regrouper des données à une vitesse élevée. Les tâches de reconnaissance vocale ou de reconnaissance d'images peuvent prendre quelques minutes au lieu de quelques heures, par rapport à l'identification manuelle par des experts humains. L'un des réseaux neuronaux les plus connus est l'algorithme de recherche de Google.

9. Resultats et disscusions

Sous python grâce à la librairie « Scikit-Learn » et une boucle qu'on a créer on a pu exécuter tous les modèles qu'on a visé d'utiliser tout en définissant au préalable la fonction accuracy qui va nous retourner l'accuracy des modèles.

```
models = [
    LogisticRegression(),
    SVC(),
    MLPClassifier(),
    RandomForestClassifier(),
    xgboost.XGBRFClassifier(max_depth=5),
    KNeighborsClassifier(n_neighbors=8, p=2)
]
```

```
accuracy=[]
for model, name in zip(models,names):
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print('Confusion matrix of ',name)
    print(confusion_matrix(y_test, y_pred))
    acc_score = accuracy_score(y_test, y_pred)
    print('Accuracy score is ',acc_score)
    accuracy.append(acc_score)
```

On obtient ainsi les matrices de confusion des modèles qui stipulent les taux de bon classement ci-dessous

	Model	Accuracy
3	Random Forest	0.931366
4	XGBoost	0.894288
5	KNeighborsClassifier	0.776743
1	Support Vector Machine	0.752446
0	Logistic Regression	0.725939
2	Neural Network	0.473967

On remarque que le modèle de random forest est le modèle le plus puissant en terme de prédiction suivi de XGboost.

Les modèles du plus proche voisin, SVM et la regression logistique ont enregistré un bon taux de précision à l'encontre du modèle de reseaux de neurones.

Pour améliorer le modèle de réseau de neurones on a essayé de changer les hyper paramètres mais le temps d'exécution était très long. Donc on opté pour la non considération de cette partie dans notre travail .

Conclusion :

Au terme de ce travail, nous avons abouti à des modèles de crédit « scoring » adapté au prêt personnel, et qui permet de prédire les personnes qui feront défaut ou non à travers leurs caractéristiques liées au profil ainsi qu'au crédit. Sur la base de données disponibles, nous avons appliqué la méthode statistique de la régression logistique ainsi que tous les autres modèles cités avant.

On a abouti à ce que le modèle obtenu par le Random forest est le plus performant suivi directement par le XGboost, en revanche le modèle de réseau de neurones était le moins performant. En effet on a essayé de changer les hyper paramètres pour plusieurs modèles mais on ne les a pas traitées finalement car les algorithmes on pris beaucoup de temps pour l'exécution.