RAPPORT D'AVANCEMENT

Prédiction du passage à l'acte suicidaire grâce à la science des données

Master 2 Ingénierie de la décision et Big Data

Sous la direction de Jean-Marie Marion

BELLAY Guillaume

Université Catholique de Bretagne Nord

Superviseur : Virginie Jacob Alby

Faculté des Sciences

Institut de Mathématiques Appliquées

Année universitaire : 2020-2021

UCO ANGERS
UNIVERSITÉ CATHOLIQUE DE L'OUEST

3 place André Leroy | BP 10808- 49008 Angers cedex 01

CHARTE DE NON PLAGIAT

Protection de la propriété intellectuelle

Tout travail universitaire doit être réalisé dans le respect intégral de la propriété intellectuelle d'autrui. Pour tout travail personnel, ou collectif, pour lequel le candidat est autorisé à utiliser des documents (textes, images, musiques, films etc.), celui-ci devra très précisément signaler le crédit (référence complète du texte cité, de l'image ou de la bande-son utilisés, sources internet incluses) à la fois dans le corps du texte et dans la bibliographie. Il est précisé que l'UCO dispose d'un logiciel anti-plagiat dans lms.uco.fr, aussi est-il demandé à tout étudiant de remettre à ses enseignants un double de ses travaux lourds sur support informatique.

Cf. « Prévention des fraudes à l'attention des étudiants »

Je soussigné, Guillaume Bellay, étudiant en Master 2 Ingénierie de la décision et Big Data, m'engage à respecter cette charte.

Fait à Guingamp, le 8/02/2021

Guillaume Bellay

Sommaire

Introduction

Le suicide est un acte de contestation maximale de l'existence. Le passage à l'acte suicidaire est le résultat d'un processus psycho-social. La découverte de ce processus suicidaire permet non seulement d'atténuer les souffrances mentales mais aussi de sauver des vies. Le geste suicidaire provoque entre 800000 et 1 million de morts dans le monde chaque année. Il s'agit d'un problème de santé publique mondial qui provoque plus de décès que les homicides et les guerres réunis.

Il existe de nombreux facteurs de risque liés au comportement suicidaire. Ceux-ci s'appliquent à l'échelle d'une population et non à celle de l'individu. Ces facteurs de risque appartiennent à plusieurs sphères : psychiatrique, psychologique, environnementale, sociale, familiale, biologique, historique et conjoncturelle. Les événements violents, la perte d'emploi, l'isolement, la perte d'un proche ou encore le sexe et l'âge sont des facteurs de risque. Il est important de préciser qu'aucune de ces variables n'est spécifique au suicide et n'est suffisante pour expliquer le geste suicidaire.

Les Côtes d'Armor étant le territoire le plus touché en France par le taux de suicidés et de suicidants, le département de psychologie de l'UCO Bretagne Nord implantée à Guingamp a lancé un programme de recherches-actions pour œuvrer à une meilleure compréhension de la psychopathologie de ce phénomène. L'objectif final est de créer un questionnaire de prévention du suicide sur tablettes. Monique Seguin, professeure au Québec, a mis à la disposition de l'UCO Bretagne Nord une banque de données unique au monde sur ce thème. Cette banque de données contient les réponses à des questionnaires portant sur les trajectoires de vie. Ces questions sont divisées en différentes sections : sociodémographique, antécédents familiaux, tentatives de suicide et trajectoires.

Mon travail a consisté à déterminer quels sont les facteurs de vulnérabilités déterminants dans le passage à l'acte suicidaire. J'ai travaillé avec Virginie Jacob Alby enseignante chercheur en psychologie à l'UCO Bretagne Nord, Charles-Edouard Notre Dame psychiatre au CHU de Lille et Jonathan Daeden data scientist chez MyDataModels.

J'ai procédé dans un premier temps à la préparation des données puis j'ai testé plusieurs algorithmes d'apprentissage automatique.

1. Préparation des données

1.1 Description du jeu de données

Notre jeu de données contient 586 lignes (1 ligne par individu) et 18827 colonnes.
Toutes nos variables sont de type entier ou réel. Les variables de type chaine de caractère présentées dans le dictionnaire des données (Annexe) étaient absentes.
Parmi les 18827 variables, 18300 d'entre elles représentent les expériences de vie adverse (elles sont surlignées en jaune, Annexe). Une expérience de vie adverse (EVA) peut être rencontrée à n'importe quel âge (0-99 ans). Pour chaque EVA nous avons donc 100 colonnes.
Parmi les 527 variables ne représentant pas des EVA, seulement 161 d'entre elles ont été retenues (elles sont surlignées en rouge, Annexe). Ces 161 variables étaient les plus importantes du point de vue du clinicien. Sur ces 161 variables, nous avons 11 variables numériques, 59 variables catégorielles et 91 variables binaires.
Notre variable à prédire est la variable « GroupeAppartenance » : est-ce que l'individu est vivant sans antécédent de tentative de suicide ? Est-ce qu'il a tenté de se suicider ? Est-ce qu'il est décédé ?
Dans notre jeu de données, nous pouvons observer la présence des valeurs 222, 888 et 999. Ces valeurs représentent une absence de réponse de l'individu. D'autre part, notre jeu de données contient de nombreuses valeurs manquantes (NAN). Pour les variables représentant des EVA, les 99 représentent des données manquantes non-aléatoires en raison du décès de l'individu (individus morts) tandis que les 77 représentent des données manquantes non-aléatoires en raison du fait que la date est ultérieure à l'entretien (individus vivants).

1.2 Nettoyage des données

Pour le nettoyage des données, je me suis servi de plusieurs variables qui initialement n'étaient pas conservées par le clinicien. Le nettoyage des données s'est déroulé en plusieurs étapes :

Etape 1 : Suppression des individus avec une valeur pour la variable « GroupeAppartenance » différente de 0,1 ou 2. Il nous reste 572 individus.

Etape 2 : Remplacement des valeurs aberrantes par NAN. Par exemple, pour toute variable binaire, les valeurs différentes de 0 ou1 sont remplacées par NAN.

Etape 3 : Remplacement des valeurs NAN par 0 pour les variables 'med1_code', 'med2_code', 'med3_code', 'med4_code', 'med5_code', 'med6_code', 'mv1_code', 'mv2_code', 'mv3_code', 'mv4_code', 'mv5_code', 'mv6_code' quand l'individu ne prend pas de médicament. Autrement dit, le type de médication est égal à « Aucun » lorsque l'individu ne prend pas de médicament. Pour savoir si un individu ne prend pas de médicament, nous pouvons regarder les valeurs des variables « med1_prise », « med2_prise », « med3_prise », « med4_prise », « med5_prise », « med6_prise », « mv1_prise », « mv2_prise », « mv3_prise », « mv4_prise », « mv5_prise », « mv6_prise ». Dans le cas où la valeur est différente de 0 ou 1, c'est que l'individu ne prend pas de médicament.

Etape 4 : Remplacement des valeurs NAN par 0 pour les variables 'men_mere' et 'men_pere' quand l'individu a sa mère ou son père décédé. Autrement dit, un individu dont le père (respectivement la mère) est décédé(e) ne vit pas avec son père (respectivement sa mère). Pour savoir si le père (respectivement la mère) de l'individu est décédé(e), il nous suffit de regarder les valeurs de 'père_vie' (respectivement 'mere_vie').

Etape 5 : Remplacement des valeurs NAN par 1 pour les variables 'mere_vie' et 'père_vie' quand l'individu vit avec sa mère ou son père. Autrement dit, un individu qui vit avec son père (respectivement sa mère) a donc son père (respectivement sa mère) vivant(e). Pour savoir si un individu vit avec son père (respectivement sa mère), il nous suffit de regarder les valeurs de 'men_pere' (respectivement 'men_mere').

Etape 6 : Remplacement des valeurs NAN par 14 pour la variable 'prof_code' quand l'individu n'a pas d'emploi rémunéré. Autrement dit, un individu qui n'a pas d'emploi rémunéré est sans emploi. Pour savoir si un individu n'a pas d'emploi rémunéré, il nous suffit de regarder les valeurs de 'prof_emploi'.

Etape 7 : Pour les variables 'enfants_nb', 'fratrie_nb', 'adopte', 'divorce', 'fam_accueil' et 'separation', les valeurs NAN, 222, 888 et 999 ont été remplacées par 0.

Etape 8 : Remplacement des valeurs 222, 888, 999 par NAN.

Etape 9 : Suppression des variables avec seulement des valeurs NAN. 16 variables ont été supprimées : 'Communauté', 'mere_vie_nb', 'père_vie_nb', 'separation_nb', 'sep_parent_nb', 'sep_age_nb', 'divorce_nb', 'divorce_age_nb', 'SZmereP', 'SZpereC', 'SZpereP', 'SZfrereC', 'SUI_2et3', 'SUI_2et3_qte', 'TS_2et3', 'TS_2et3_qte'.

1.3 Imputation des données manquantes

L'imputation des données manquantes peut être décomposée en trois parties.

1- Imputation des données manquantes pour les variables représentant les EVA
Les valeurs NAN, 99 et 77 ont été remplacées par 0. Le traitement de ces variables sera développé au point 1.4.

2- Imputation des données manquantes pour les 125 premières variables
Après le nettoyage des données, il nous reste 145 variables (sans compter les variables représentant les EVA). Les 20 dernières sont les variables « fardeau » qui seront traitées dans un deuxième temps.
Pour l'imputation des 125 variables j'ai utilisé la librairie MissingValuesHandler.
MissingValuesHandler utilise l'algorithme Random Forest (Forêts aléatoires) pour remplacer les valeurs manquantes.
Petit rappel sur l'algorithme Random Forest : on effectue un apprentissage sur de multiples arbres de décision sur des sous-ensembles de données légèrement différents. Cela nous permet de réduire la variance. Cet algorithme donne de bons résultats surtout en grande dimension, c'est simple à mettre en œuvre et il y a peu de paramètres.
Le type de Random Forest (régression ou classification) est traité automatiquement. Si la variable à prédire est numérique, alors on utilise RandomForestRegressor et si elle est catégorielle alors on utilise RandomForestClassifier. Dans notre cas, ce sera RandomForestClassifier.
L'algorithme n'impute pas toutes les valeurs manquantes dès le premier coup. Celui-ci doit tourner plusieurs fois (dans notre cas 5 fois) avant qu'il y ait convergence.

3- Imputation des données manquantes pour les variables « fardeau »
Tout d'abord, j'ai regroupé les variables « fardeau » avec les 125 variables qui sont cette fois-ci sans valeurs manquantes. Puis j'ai utilisé une nouvelle fois la librairie MissingValueshandler.

L'imputation des 145 variables ne pouvait être réalisée en une seule fois. Avec ces 145 variables, l'algorithme n'arrivait pas à remplacer toutes les valeurs manquantes (non-convergence).

1.4 Traitement des variables représentant les EVA

Nous avons 18300 variables représentant les EVA. Il ne semble pas pertinent d'avoir un modèle avec autant de variables. Cela rendrait le modèle trop complexe et nous aurions sans aucun doute un surapprentissage (overfitting). Nous avons donc cherché à réduire le nombre de dimensions sans toutefois perdre en interprétabilité.
Pour cela, j'ai sommé les valeurs de ces variables de trois façons :

1- La somme sur toutes les variables. Ainsi, nous pouvons savoir pour chaque individu le nombre d'EVA de tout type qu'il a rencontré au cours de sa vie (0-99 ans). Nous obtenons donc une colonne.
2- La somme par type d'EVA. Ainsi, nous pouvons savoir pour chaque individu le nombre d'EVA qu'il a rencontré au cours de sa vie (0-99 ans) pour chaque type d'EVA. Il y a 183 types d'EVA donc nous obtenons 183 colonnes.
3- La somme par tranche d'âge. Il y a 20 tranches d'âge : 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75-79, 80-84, 85-89, 90-94, 95-99. Ainsi, nous pouvons savoir pour chaque individu le nombre d'EVA de tout type qu'il a rencontré pour chaque tranche d'âge. Nous obtenons 20 colonnes.

De cette façon nous obtenons 204 colonnes au lieu de 18300.

1.5 Création de notre jeu de données final

J'ai rassemblé les 145 variables sans valeurs manquantes avec les 204 colonnes créées. La variable « GroupeAppartenance » est déplacée en dernière position car il s'agit de notre variable à prédire. Enfin les variables avec une seule valeur ne sont pas conservées.
Nous obtenons un jeu de données avec 572 lignes et 337 colonnes.
Si on ne prend pas en compte notre variable à prédire, nous avons 36 variables catégorielles, 224 variables numériques et 76 variables binaires.
171 individus appartiennent à la classe 0 (vivant, sans antécédent de suicide), 98 appartiennent à la classe 1 (tentative de suicide) et 303 appartiennent à la classe 2 (décès suicide).

2. Entrainement de modèles d'apprentissage automatique

2.1 Premier modèle avec LightGBM

Pour notre premier modèle, j'ai utilisé LightGBM (Light Gradient Boosting Machine).
Le Gradient Boosting est un type d'apprentissage ensembliste. L'apprentissage ensembliste utilise plusieurs algorithmes d'apprentissage pour obtenir de meilleures prédictions. Dans notre cas, ces ensembles sont construits à partir d'arbres de décision. Les arbres de décision sont ajoutés un à un à l'ensemble et corrigent les erreurs de prédiction commises par les arbres de décision ajoutés auparavant.

Avant de lancer notre modèle, il nous faut encore une fois réaliser une préparation des données. Cette préparation ne doit être réalisée que sur le jeu de données d'entrainement, pour éviter toute « fuite des données » (data leakage). Pour cela j'ai créé un pipeline avec Scikit-learn. Ce pipeline est composé de plusieurs étapes :

1- Préparation des données
   A- Variables catégorielles : encodage des variables
   B- Variables numériques :
      - Suppression des variables avec une faible variance
      - Normalisation
2- Modèle LightGBM

Pour les variables catégorielles, j'ai utilisé la fonction OneHotEncoder de Scikit-learn. Pour les variables numériques, j'ai utilisé les fonctions VarianceThreshold et MinMaxScaler de Scikit-learn. Toutes les variables avec une variance inférieure à 0.2 sont supprimées. Il n'y a pas de préparation pour les variables binaires. Celles-ci sont tout de même conservées.
Il nous faut aussi prendre en compte le déséquilibre des classes. En effet, la classe 2 est dominante. 171 individus appartiennent à la classe 0, 98 appartiennent à la classe 1 et 303 appartiennent à la classe 2. Ce déséquilibre n'est pas très fort mais il doit tout de même être traité. Un paramètre de notre fonction LightGBM nous permet de résoudre ce problème. Il s'agit du paramètre « class_weight ». En précisant « class_weight='balanced' », nous appliquons des poids inversement proportionnels aux effectifs des classes.

Formule pour calculer les poids : $\frac{n_{samples}}{n_{classes} \cdot n\_samples\_with\_class}$

Donc dans notre cas, nous avons :

Poids_classe0 = 572 / (3 x 171) = 1.11

Poids_classe1 = 572 / (3 x 98) = 1.95

Poids_classe2 = 572 / (3 x 303) = 0.63

Une fois le pipeline créé, nous pouvons l'évaluer grâce à une validation croisée. Pour cela, j'ai utilisé la fonction cross_val_score de Scikit-learn. Il existe plusieurs stratégies pour séparer les jeux d'entrainement et de test. Ici j'ai utilisé la stratégie « StratifiedKFold » (Scikit-learn). Avec « StratifiedKFold », la répartition des classes est la même dans les jeux d'entrainement et de test.

Résultats :

Nous obtenons un taux de bonnes réponses (accuracy) de 82.35% (validation croisée).

Matrice de confusion pour le jeu de test :

Actuel / Prédiction	0	1	2
0	29	0	2
1	0	14	7
2	6	8	49

Rapport de classification pour le jeu de test :

Classe	Précision	Rappel	F1-score
0	0.83	0.94	0.88
1	0.64	0.67	0.65
2	0.84	0.78	0.81

Rappel des métriques :

Précision = TP / (TP + FP)

La précision nous donne la proportion de positifs qui sont réellement des positifs.

Rappel = TP / (TP + FN)

Le rappel nous donne la proportion de positifs qui ont bien été prédits comme tels.

F1-score = 2 x $\frac{Précision \cdot Rappel}{Précision + Rappel}$

Au vu de ces résultats nous pouvons dire que notre modèle a plus de mal à prédire la classe 1.

Dans notre cas, se baser uniquement sur une métrique comme l'exactitude (accuracy) est une erreur car nos classes n'ont pas les mêmes effectifs. Prédire correctement la classe dominante et « se rater » pour la classe minoritaire peut tout de même vous assurer une exactitude (accuracy) correcte.

Il est plus intéressant de se baser sur une autre métrique : l'aire sous la courbe (ROC AUC).

Cet indicateur ne se concentre pas sur les erreurs de prédiction mais plutôt sur la capacité d'un modèle à bien séparer les classes. L'aire sous la courbe est comprise entre 0.0 et 1.0. Plus cette aire est importante, plus le modèle est performant.

La fonction roc_auc_score de Scikit-learn nous permet de calculer l'aire sous la courbe.

En précisant « multi_class='ovo' », nous pouvons comparer les classes deux à deux.

En précisant « average='weighted' », nous pouvons calculer cette métrique pour chaque label puis faire la moyenne pondérée par leurs effectifs.

Résultat: roc_auc_score = 0.9275

Suite du stage

Voici ce que j'aimerai faire avant la fin de mon stage :

- Déterminer les variables qui contribuent le plus au modèle (fonction permutation_importance de Scikit-learn et/ou SHAP)
- Comparer LightGBM à d'autres algorithmes
- Optimisation des hyperparamètres (Auto-sklearn et/ou pyautoweka et/ou FLAML)
- Réaliser une classification binaire, avec les classes 1 et 2. Cela permettrait de mieux comprendre ce qui différencie une personne qui tente de se suicider d'une personne qui décède.
- Participer au colloque du 24 Juin et écrire un article avec Virginie Jacob Alby.

Bibliographie

Jérémie Vandevoorde. 2018. *Evaluer le risque de suicide.*

Fabrice Jollant et Philippe Courtet. 2010. *Modèle neuroanatomique et homéostatique des conduites suicidaires.*

Monique Séguin, Guy Beauchamp, Marie Robert, Mélanie DiMambro et Gustavo Turecki. 2015. *Developmental model of suicide trajectories*.

Python documentation. 2021. En ligne https://docs.python.org/3/, consulté le 29 avril 2021.

Librairie Pandas. 2021. En ligne https://pandas.pydata.org/docs/, consulté le 29 avril 2021.

Librairie Scikit-learn. 2021. En ligne https://scikit-learn.org/stable/index.html, consulté le 29 avril 2021.

Librairie MissingValuesHandler. 2020. En ligne https://github.com/YA26/MissingValuesHandler, consulté le 29 avril 2021.

LightGBM documentation. 2021. En ligne https://lightgbm.readthedocs.io/en/latest/index.html , consulté le 29 avril 2021.

SHAP documentation. 2018. En ligne https://shap.readthedocs.io/en/latest/, consulté le 29 avril 2021.

Auto-sklearn documentation. 2021. En ligne https://automl.github.io/auto-sklearn/master/, consulté le 29 avril 2021.

Annexes