Université Paul Valéry - Montpellier 3

UFR 6 - Éducation & Sciences pour les LLASHS

Master MIASHS

Prédiction des comportements suicidaires

RENDU 7 - MODÉLISATION

Équipe :
Lisa BÉTEILLE
Matéo CALSACY
Jean CHABANOL
Anamé ROUMY
Laura SÉNÉCAILLE
Célia TEYSSIER

2021/2022

Encadrants :
Sandra BRINGAY
Pierre LAFAYE de MICHEAUX
Florian LOMBARDO

SOMMAIRE

TABLE DES FIGURES ----------3

INTRODUCTION----------4

MODÉLISATIONS DU 1ER SEMESTRE ----------5

PARTIE 1 - DESCRIPTION DES ALGORITHMES----------6

I. K-Means **6**

a. Définition générale 6

b. Fonctionnement du K-Means 7

II. SVM **8**

a. Définition générale 8

b. Fonctionnement du SVM 8

III.Random Forest **11**

a. Définition générale 11

b. Fonctionnement du Random Forest 11

PARTIE 2 - LA VALIDATION CROISÉE ----------13

PARTIE 3 - LES ALGORITHMES SUR NOS DONNÉES ----------14

I. K-Means **14**

II. SVM **14**

III.Random Forest **15**

MODÉLISATIONS DU 2ÈME SEMESTRE ----------17

I. Description de BERT **18**

II. Fonctionnement du modèle de BERT **19**

III.BERT sur nos données **22**

CONCLUSION----------23

SOURCE ----------24

TABLE DES FIGURES

Figure 1 : Représentation d'un K-means (source) 6

Figure 2 : Exemple de représentation séparable 9

Figure 3 : Exemple de représentation non séparable 9

Figure 4 : Transformation de 2D à 3D (transformation SVM) 10

Figure 5 : Exemple d'arbre de décision avec la librairie sklearn du jeu de données iris (source) 11

Figure 6 : Représentation d'une validation croisée (source) 13

Figure 7 : Arbre de décision sur nos données 15

INTRODUCTION

Notre projet consiste à appliquer plusieurs algorithmes qui permettront de déterminer si un texte peut avoir été écrit par une personne risquant de faire une tentative de suicide.

Les commanditaires souhaitent un algorithme qui identifient un comportement suicidaire sur différents types de données : Wikipédia, tweets et forum Reddit. Nous avons donc constitué des corpus qui font références à un suicide ou non.

L'objectif de ce projet est donc de commencer par tester des algorithmes classiques par classification binaire avec le SVM, le random forest et le K-means. Puis de tester l'hypothèse du transfert. On apprend le modèle sur un jeu de données où on est sûrs des étiquettes labellisées avec Wikipédia, et on observe si le modèle fonctionne bien quand on le teste sur des sources différentes.

Les résultats de ces modèles sont détaillés dans le rapport 9 qui concerne l'explication et l'interprétation des résultats.

Au premier semestre, nous avons commencé la classification des données avec des modèles assez simples d'apprentissage automatique, avant d'aller vers des modèles d'apprentissage profond au second semestre.

MODÉLISATIONS DU 1ER SEMESTRE

Après avoir effectué toute la partie sur l'exploration des données et le prétraitement des données (cf. compte rendu 3), nous avons mis en place des modèles de classification supervisée.

Pour savoir à quelle catégorie appartiennent nos données, nous avons ajouté des variables Vrai et Faux qui correspondent à la catégorie suicide et non suicide dans chaque corpus.

Suite au nettoyage et à l'harmonisation des données dans le but d'avoir des jeux de données équilibrés, nous avons obtenus plusieurs corpus :

- Données Wikipédia : 900 suicide et 900 non suicide.
- Données Reddit : 4723 suicide et 4723 non suicide.
- Données Twitter : 2 456 suicide et 2 456non suicide.

Durant la première partie de l'année, nous avons utilisé des modèles d'apprentissage automatique classique, que nous maîtrisons et que nous avons vu lors d'un des enseignements du master 'Classification supervisée et non supervisée'.

Après plusieurs temps de réflexion et de par nos connaissances, nous nous sommes posés la question : *quel est le meilleur algorithme d'apprentissage à utiliser pour notre tâche* ?
C'est ainsi que nous avons décidé d'appliquer 3 modèles, qui sont tous adaptés à nos données, le K-means, le Random Forest, et le SVM.

Ces modèles nous permettront d'obtenir des premiers résultats au travers des métriques de classification telles que l'accuracy, le rappel, la précision et le F1-score.

Dans un premier temps, au travers de ces modèles nous avons cherché à entraîner et tester nos données sur chacun des jeux de données que nous avons à notre disposition. Donc dans un cas comme celui-ci, on devrait obtenir des scores de précision assez élevés.

En effet, il s'agissait d'une tâche extrêmement facile pour chaque jeux de données. Par exemple, nous cherchons à savoir si le tweet était bien prédit à suicide s'il s'agissait d'un tweet suicidaire.

Nous allons donc vous expliquer pourquoi nous avons choisis ces 3 classifieurs au travers de la définition et du fonctionnement de chacun d'eux. Nous parlerons ensuite du principe de la validation croisée, méthode que nous avons employée lors de l'exécution de nos modèles. Puis nous finirons cette première partie sur l'exécution des algorithmes de classifications sur nos données.

PARTIE 1 - DESCRIPTION DES ALGORITHMES

I. K-Means

a. Définition générale

Le k-means (ou K-moyennes) correspond aux plus proches voisins. Il est l'un des algorithmes de clustering les plus répandus. Il permet d'analyser un jeu de données caractérisées par un ensemble de descripteurs afin de regrouper les observations qui paraissent "similaires" en clusters. C'est-à-dire un ensemble de points qui ont été déterminés comme étant les plus proches entre leurs voisins (voir Figure 1).

Voici une représentation de données analysées avec un k-means :

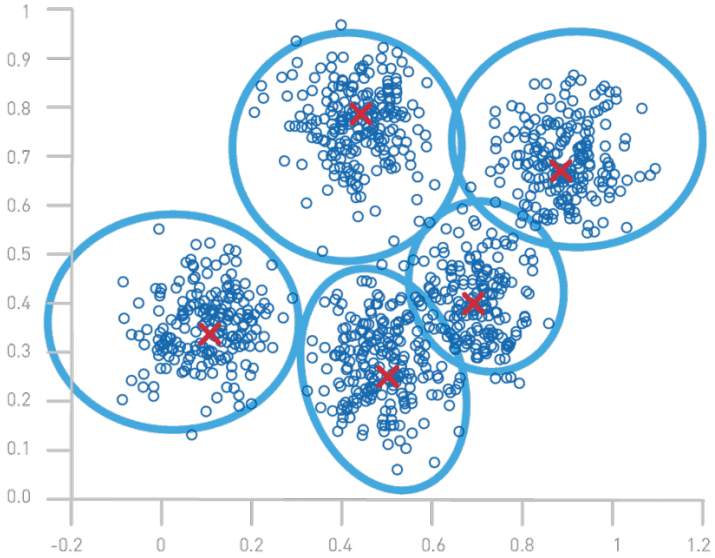


Figure 1 *: Représentation d'un K-means (source)*

Équation mathématique :

On considère un ensemble de points ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), on cherche à faire une partition des n points en k ensembles $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ ($k \leq n$) en minimisant la distance entre les points à l'intérieur de chaque partition :

où $\boldsymbol{\mu}_i$ est le barycentre des points dans S_i.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

Autrement dit, on minimise la somme des distances entre chaque individu et le centroide (centres des clusters) en sachant que le **choix initial des centroïdes conditionne le résultat final**.

b. Fonctionnement du K-Means

Le principe de l'algorithme est d'avoir en entrée le nombre de clusters à former (K) et le Training Set (la matrice de données). On choisit ensuite aléatoirement k points. Ces points représentent les centres des clusters.
Après avoir initialisé ces centroïdes, l'algorithme du *K-means* alterne plusieurs fois les deux étapes ci-dessous pour optimiser les centroïdes et leurs groupes :

1. Regrouper chaque objet autour du centroïde le plus proche
2. Recalculer chaque centroïde selon la moyenne des descripteurs de son groupe

—> Après toutes ces itérations, on obtient un découpage optimal : on a atteint la convergence du modèle.

La condition d'arrêt du modèle peut s'obtenir de deux manière différentes :

Par la présence d'un nombre d'itérations fixé à l'initialisation du modèle. Le modèle s'arrêtera après avoir atteint cette limite même si les clusters ne sont pas bien formés.

Par une stabilisation des centroïdes des clusters, c'est-à-dire qu'ils ne bougent plus lors des itérations.

On utilise le K-means pour regrouper n'importe quels éléments similaires dans des clusters. La seule condition d'utilisation est qu'ils soient encodés dans une matrice de données.

Avantages du K-mean :

On pourrait dire du K-means qu'il est simple, rapide et facile à interpréter comme nous l'avons vu avec son fonctionnement.

Les hyperparamètres du K-means :

Les hyperparamètres du K-means sont très simples à comprendre, en voici quelques uns, les plus importants :

- n_clusters : correspond au nombre de clusters que nous désirons, correspond au k.
- random_state : Détermine le nombre aléatoire pour l'initialisation du centroïde.

II. SVM

a. Définition générale

Les machines à vecteurs de support, appelé aussi SVM, sont un ensemble de techniques d'apprentissage supervisé employées dans la résolution des problèmes de discrimination (technique statistique qui vise à prédire l'affiliation à des groupes fixés d'un ensemble d'observations) et de régression.

En d'autres termes, le SVM a pour but de trouver un hyperplan qui divise de manière optimale un ensemble de données en deux parties en fonction des observations. L'idée est donc de projeter les données dans un espace à grande dimension pour rendre les observations séparables.

Equation mathématique :

On sait donc que les SVM sont dans la plupart des cas choisis pour résoudre des problème de discrimination avec pour objectif de classer les observations en prédisant la valeur numérique d'une variable.
Dans le cas d'un problème de discrimination binaire, on prend en compte une fonction h qui à un

$$y = h(x)$$

vecteur d'entrée x qui correspond à y :
Nous sommes dans un cas binaire, donc $y \in \{ -1, 1\}$ et l'espace X qui comprend le vecteur d'entrée X peut s'écrire de la manière suivante : $X = \mathbb{R}^N$

b. Fonctionnement du SVM

Le SVM a un fonctionnement bien particulier sur le principe de mapping des données, c'est-à-dire qu'il va chercher à mettre en correspondance les données ayant les mêmes observations en les classant. Et cela même si les données ne sont pas séparables sur un plan linéaire.

Dans un premier temps, partons du principe que nos données sont linéairement séparables. Le but du SVM dans ce cas là est de chercher une frontière séparatrice entre les données, et de ce fait, il va utiliser le principe de la marge maximale. La marge est la distance entre l'hyperplan et les échantillons les plus proches.

La frontière choisie doit maximiser la distance avec les points les plus proches de la frontière en question. Après plusieurs étapes d'entraînement, le SVM sait où placer la frontière pour les données testées.

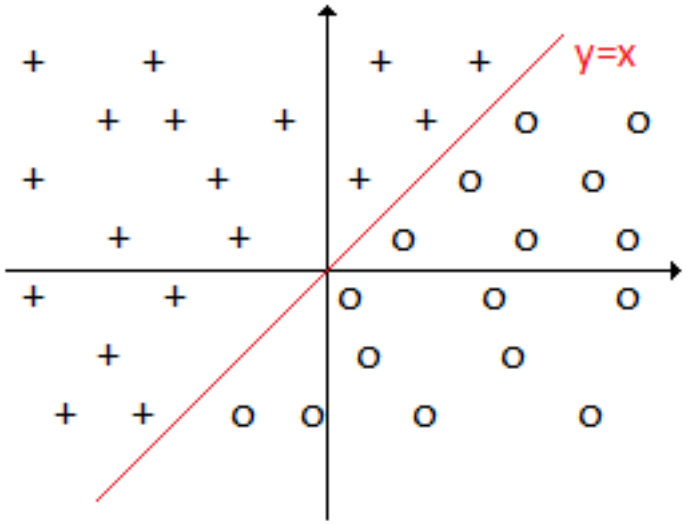


Figure 2 *: Exemple de représentation séparable*

Dans la figure 2, on remarque bien que les données étaient séparables, il s'agit d'un cas simple du SVM. Car la droite y=x est linéaire.

Mais que se passe-t-il lorsque les données ne sont pas linéairement séparables, comme sur la figure 3 ?

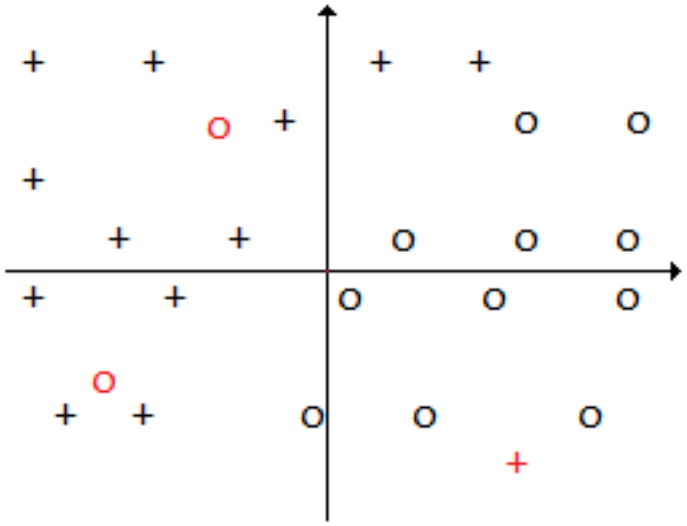


Figure 3 *: Exemple de représentation non séparable*

On utilise dans ce cas une fonction noyau en plus de la méthode de la marge maximale. Le SVM prend en charge plusieurs types de noyaux et dans ce cas, on utilise les noyaux de types Polynomial ou Sigmoïde.

On procède à une transformation des données pour les rendre séparables, et de ce fait, on passe de dimension 1 en dimension 2 grâce à la fonction noyau (voir Figure 4).

On peut résoudre cela en utilisant la fonction cube comme fonction noyau. Et donc re projeter les données avec cette fonction.

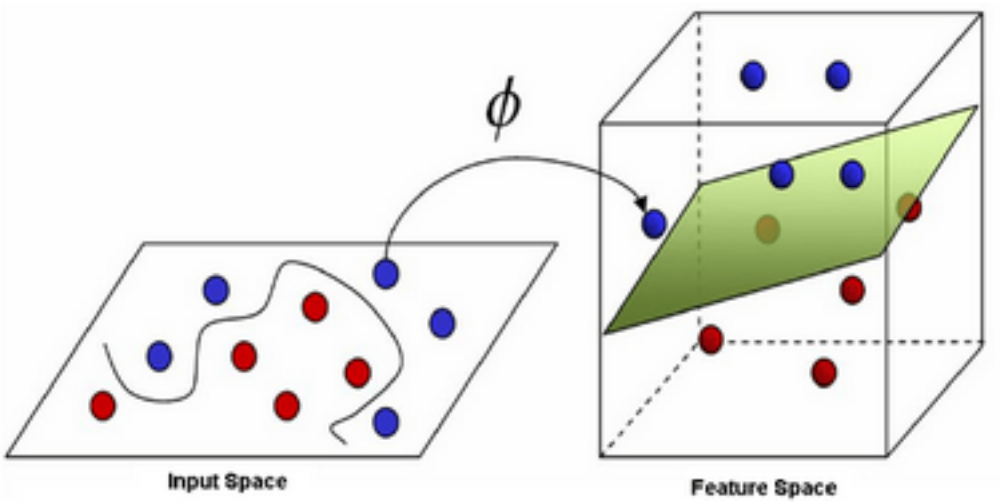


Figure 4 : *Transformation de 2D à 3D (transformation SVM)*

Pré-conditions et avantages du SVM :

Le principal avantage du SVM comparé aux autres modèles est qu'il existe des méthodes d'hyper-paramètres par défaut, que l'on peut utiliser dans le cadre d'une classification par exemple. De par ce fait, le nombre d'hyper paramètres est très restreint, et nous avons juste besoin de renseigner la technique de régularisation des données et le choix du noyau.

Les hyperparamètres du SVM :

Sous la librairie Scikit-Learn, le SVM a un certain nombre d'hyper paramètres :

- *kernel* : type de noyau à utiliser (rbf, linear, poly)
- *C* : paramètre de régularisation et de précision. Ce paramètre permet d'éviter le sur-apprentissage
- *degree* : paramètre qui définit le degré du polygone. Il est uniquement utilisé si le noyau 'poly' est choisi
- *probability* : Il s'agit d'un paramètre booléen. S'il est à True, alors le modèle renverra une évaluation pour chaque prédiction
- *gamma* : paramètre qui définit l'influence des points. Il est utilisé uniquement si le noyau 'rbf' est choisi

III.Random Forest

a. Définition générale

Le random forest est un assemblage de plusieurs arbres de décision, qui fonctionnent de manière indépendante. Chacun produit une estimation et c'est l'assemblage des arbres de décision et de leurs analyses, qui va donner une estimation globale.

b. Fonctionnement du Random Forest

Un random forest, comme expliqué précédemment, est une combinaison de plusieurs arbres de décision.

Pour introduire le random forest, nous avons donc besoin de décrire le principe d'un arbre de décision.
Les arbres de décision sont formés par une série d'étapes simples dans le but de fournir des probabilités, et des réponses liées à la prise d'une décision.
Le but de ce processus est de répondre successivement à des conditions à vérifier, chaque condition correspond à une caractéristique de notre jeu de données. À chaque étape, une branche est ajoutée en fonction de la réponse, oui ou non. Ce qui permet d'obtenir une organisation hiérarchique sous forme d'arborescence.

Ces arbres sont relativement simples à comprendre et à interpréter et cela par la visualisation claire et précise de ce que le modèle renvoie comme nous pouvons le voir sur la figure 5.

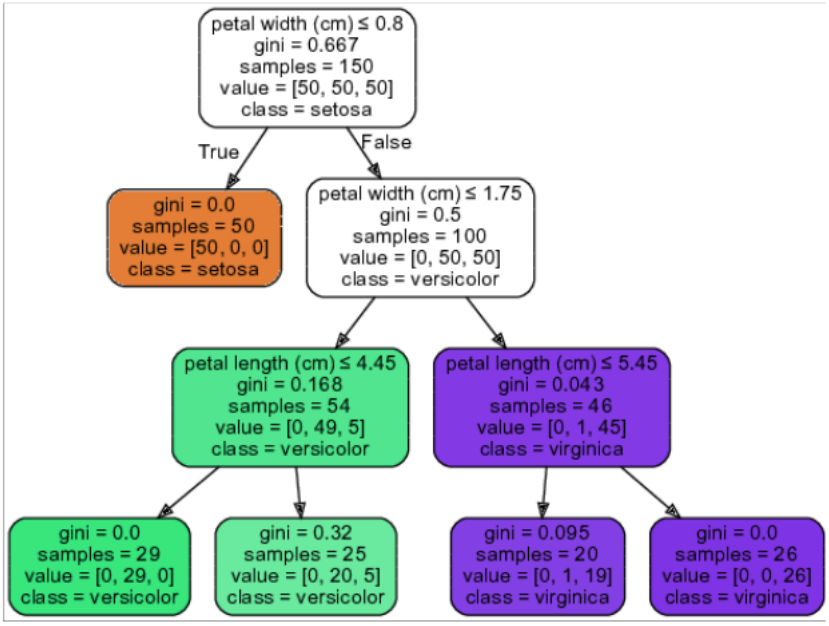


Figure 5 *: Exemple d'arbre de décision avec la librairie sklearn du jeu de données iris (source)*

Mais ces arbres contribuent à la création d'erreurs dues aux biais et à la variance. La réponse d'un seul arbre de décision est un indicateur de décision assez faible.

On peut dès à présent introduire la notion de random forest.
Le random forest est une association d'un grand nombre d'arbres de décision qui sont assemblés à la fin du processus d'apprentissage.

Nous avons trouvé une brève explication du fonctionnement du random forest qui est produite en 3 étapes :

Étape 1 - Principe du bagging : on créer plusieurs sous-échantillons aléatoires de l'ensemble de données.

Étape 2 - Création de plusieurs arbres de décision indépendant pour chaque sous-échantillon dans le but d'avoir des prédictions. L'assemblage de tous ces arbres permettra de réduire les problèmes de biais / variance des arbres de décision individuels.

Étape 3 - Prédiction des résultats. On récupère la moyenne des scores de précision de l'ensemble des arbres de décision. Cela permet d'avoir un modèle plus robuste, et d'obtenir des résultats très proches de la réalité.

Les hyperparamètres du random forest :
Le random forest a besoin de 3 hyperparamètres qui doivent être définis lors de la mise en place du modèle :

- max_depth : correspond à la taille des arbres
- n_estimators : correspond au nombre d'arbres de décision à prendre en compte
- max_features : correspond au nombre de variables aléatoires choisies à chaque mélange

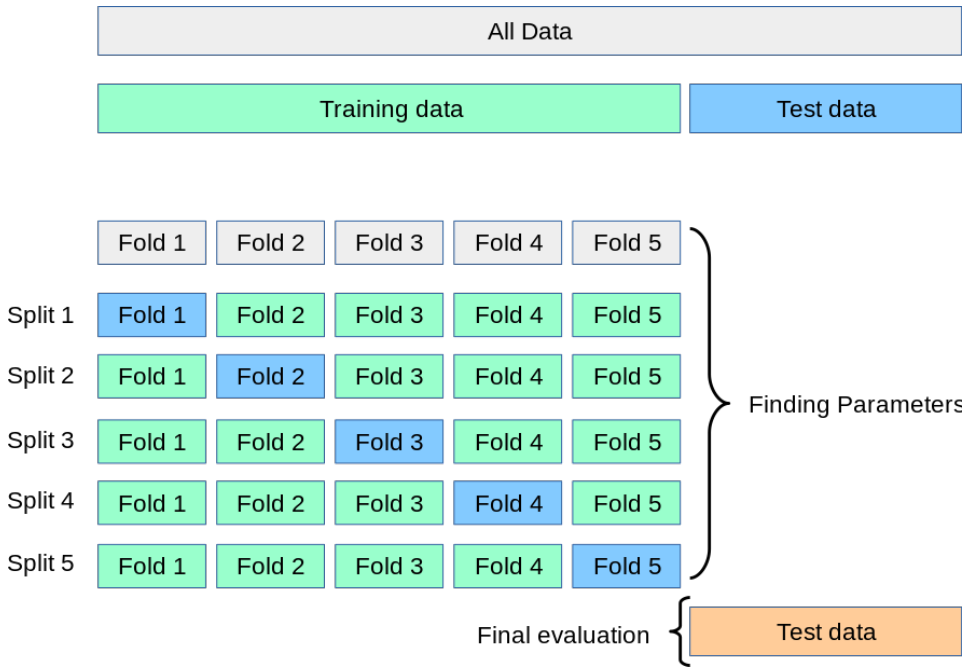
L'inconvénient du random forest :
Le seul inconvénient que l'on peut retenir de ce modèle est que plus le nombre d'arbres que l'on implémente à notre modèle est élevé plus le processus sera lent et coûteux. En effet, chaque arbre devra être analysé un après l'autre, le seul moyen de contrer cela est de réduire le nombre de caractéristiques.

PARTIE 2 - LA VALIDATION CROISÉE

Après avoir entraîné ces trois modèles de classification, nous avons effectué une validation croisée sur les données.
C'est une méthode qui va permettre de tester les performances réelles de nos modèles prédictifs.
Pour ce faire, il est indispensable de comprendre le fonctionnement du paramètre central de la méthode K-Fold, que nous avons choisi au détriment de la méthode train-val qui est plus répandue.

Le K-Fold est très populaire. Il permet de garantir que toutes les informations du jeu de données de base aient l'opportunité d'apparaître à la fois dans l'ensemble d'entraînement et de test.
On commence par déterminer un paramètre K qui correspond au nombre de groupes dans lequel l'échantillon sera divisé, généralement on choisit un K aux alentours de 5 ou 7. Ce qui permet de ne pas trop diviser le jeu de données au risque de faire du sur-apprentissage comme nous le voyons sur la figure 6.

Figure 6 *: Représentation d'une validation croisée (source)*



On commence l'initialisation en paramétrant le K en K-1. La validation du modèle est effectuée avec les K-fold restants. Le processus continue jusqu'à ce que chaque K-fold soit utilisé dans le jeu d'entraînement. Au terme, nous obtenons la moyenne des scores.

Dans Python, nous avons utilisé les deux fonctions 'cross_val_score' pour calculer l'accuracy de chaque k-fold, et 'cross_val_predict' pour la prédiction de chacun d'entre eux.

PARTIE 3 - LES ALGORITHMES SUR NOS DONNÉES

I. K-Means

Nos données ont toutes un label qui correspond au suicide ou non. C'est par la présence de ce label que nous voulons classifier de manières binaires nos données.

Nous nous sommes tournés vers l'algorithme du K-means, car c'est un modèle très simple à comprendre et il nous paraissait cohérent de l'utiliser dans le cas de nos données.

Comme le but du k-means et de trouver les plus proches voisins, ce classifieur aurait pu être utile pour répondre à la problématique. On aurait pu regrouper tous les textes étiquetés suicide entre eux et non-suicide entre eux. Mais cet algorithme découle d'une classification non supervisée, nous n'aurions pas dû l'utiliser. En effet, si cette classification est non supervisée cela signifie qu'elle n'a pas de label.

II. SVM

Dans notre cas, toutes les données que l'on a on un label qui correspond au suicide ou non. C'est par la présence de ce label que nous voulons classifier nos données de manière binaire.
Notre problématique s'apparente bien à un problème de descrimination, car on cherche à classer les tweets, reddit ou page wikipédia en fonction d'un label suicide ou non suicide, problème traité par le SVM.

Pour la mise en place du modèle SVM, nous avons effectué plusieurs étapes de prétraitement des données.
Nous avons tokenisé chaque texte de chaque corpus de données dans le but de faire des analyse sur chaque token (mot) employé lorsque l'observation est définie comme étant un suicide ou non.
Nous avons ensuite coupé nos données en jeux de train-val-test à partir du nouveau corpus de tokens dans l'optique de faire de la validation croisée.
Pour finir, nous avons fait le choix de transformer nos labels en format numérique pour que notre modèle puisse prendre les données.
Pour finir, nous avons utilisé une méthode le TF-IDF. Cette méthode est employée dans la fouille de texte et l'analyse textuelle. C'est une mesure statistique qui permet d'attribuer une importance à un terme ou un mot contenu dans un corpus par rapport à son occurrence dans le texte.

Suite à cela, nous avons fait plusieurs choix au niveau des hyper-paramètres.

- C = 1.0 - Nous avons choisi ce paramètre par défaut. Étant donné que le C correspond au paramètre de régularisation de la précision, on a choisi de rester neutre.
- kernel = 'linear' - Nous avons choisi de mettre un noyau linéaire car nos données sont séparables linéairement.

De part nos recherches et nos connaissances, on s'est rendu compte que le SVM était un choix intéressant pour répondre à notre problématique. En effet, le SVM est fréquemment utilisé dans le cas d'analyse textuelle pour la classification de texte, or ici nous cherchons à classifier des textes suivant leur nature, suicide ou non.

Malgré cela, on s'est aussi rendu compte que ce modèle n'était pas vraiment interprétable avec nos propres mots. On peut donc qualifier ce classifieur de 'boîte noire', car par exemple la transformation de la 2D à la 3D et la création de la frontière sont assez complexes et laborieuses à expliquer.

III.Random Forest

Le Random Forest est utilisé dans de nombreux domaines, mais sa principale caractéristique est la prédiction de comportements futurs par une série d'étapes.

Or ici, on l'utilise dans notre cas, avec pour objectif de s'inspirer d'une multitude d'avis sur nos observations qui traitent sur le même sujet "Est-ce que ce texte peut alerter sur un suicide ?". Si la réponse est Oui alors l'observation en question est traité comme un suicide, et inversemment si la réponse est non. Il s'agit d'un arbre de décision assez simple (voir figure 7).

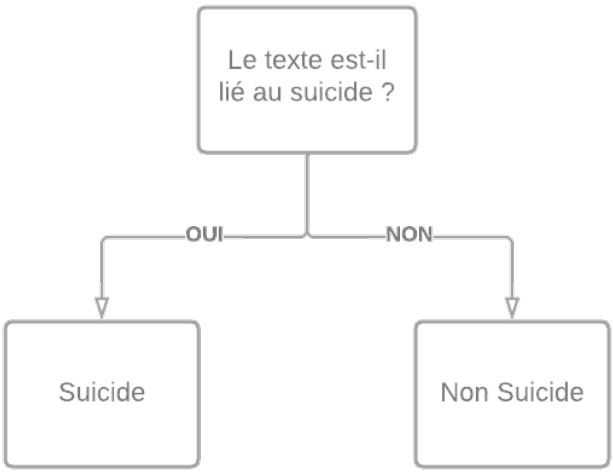


***Figure 7** : Arbre de décision sur nos données*

La différence avec les autres modèles est que celui-ci se base sur la réponse de plusieurs arbres de décision, et donc plus il y a aura d'arbre de décision, plus la réponse sera précise.

Pour la mise en place du Random Forest, nous avons effectué plusieurs étapes de prétraitement des données. Ce sont exactement les mêmes que le SVM.

Puis, nous avons eu la même réflexion au niveau des hyper-paramètres.

- max_depth = 5 - correspond à la profondeur des arbres (sous-échantillon)
- n_estimators = 100 - un nombre élevé d'arbres de décision permet d'avoir une prédiction assez proche de la réalité.

Le Random forest a été une évidence, c'est un modèle d'apprentissage assez simple à mettre en œuvre et compréhensible par son nombre peu important d'hyper paramètres. De plus, les performances de ces modèles sont généralement réalistes et précises.

MODÉLISATIONS DU 2ÈME SEMESTRE

Dans la seconde partie du semestre, on s'est dirigé vers de l'analyse textuelle et le traitement du langage naturel dans le but d'avoir des résultats plus interprétables pour pouvoir répondre à l'hypothèse générale de notre projet, 'L'observation est-elle perçu comme un suicide ?'.

Toujours dans l'optique de faire de la prédiction sur les comportements suicidaires, nous reprenons les mêmes données que nous avons prétraitées lors du semestre 1 associées avec les labels 'suicide' et 'non suicide'.

L'objectif premier du second semestre serait de pouvoir entraîner notre modèle sur les corpus de Wikipédia, qui sont des textes inclusifs, où on sait que la pertinence des données est fondée comme ce sont des personnes dont on est sûr qu'elles se sont suicidé, et de le tester sur les données reddit et twitter, où dans ce cas là nous ne sommes pas sûrs des comportements suicidaires des observations.

Après plusieurs recherches et lectures sur le sujet (https://www.quantmetry.com/blog/bert-google-ai-banc-de-test/), nous nous sommes penchés vers un modèle d'apprentissage profond du traitement du langage naturel : l'algorithme de BERT.

Nous allons donc vous expliquer pourquoi nous avons choisi cet algorithme au travers de sa définition et de son fonctionnement, puis nous terminerons cette partie avec les étapes nécessaires du modèle de BERT avec nos données.

I. Description de BERT

Parmi tous les modèles qui composent le NLP, notre approche s'est dirigée vers le modèle de BERT, un algorithme implémenté par Google.

BERT est un modèle de type Transformers. C'est-à-dire, un modèle qui se crée en effectuant un nombre d'étapes limité. À chaque étape, il applique un mécanisme d'attention pour comprendre les relations entre les mots de la phrase, quelles que soient leurs positions respectives.

- Il s'agit d'un modèle de représentation de textes écrit en langage naturel, qui a une représentation contextuelle. C'est-à-dire que dans ce modèle, un mot sera représenté en fonction de son sens et pas de manière statique, ce qui est très pertinent avec cet algorithme car suivant le contexte de la phrase un mot ne sera pas défini de la même manière.

- De plus, nous pouvons considérer ce modèle comme étant bidirectionnel. En d'autres termes, les mots qui précèdent et qui suivent le mot observés seront pris en compte dans l'interprétation de la phrase.

Il ne faut pas oublier que ce modèle découle d'un apprentissage profond sur la base de réseaux de neurones. Un réseau de neurones est un algorithme informatique qui a une composition similaire à la structure en réseau des neurones du cerveau. Il peut apprendre à partir de données et ainsi être entraîné à prédire des comportements ou des événements.

On utilise ce type de modèle dans les problèmes de classification de textes ou de prédiction de comportements qui pourrait s'apparenter à des comportements compréhensibles par le cerveau humain par exemple.

II. Fonctionnement du modèle de BERT

En quelques mots, le principe d'utilisation du modèle de BERT est le suivant : c'est un algorithme qui est déjà pré-entraîné sur une grande quantité de données, suivant les tâches que nous souhaitons effectuer nous le modifions, puis on procède à un réentraînement sur nos corpus de données. On peut le modifier en ajoutant des réseaux de neurones en sortie du modèle : cela se nomme le fine-tuning. Le processus BERT passe par deux étapes : le prétraitement et l'encodage.

Le pré-entraînement de l'algorithme est non supervisé, il ne nécessite donc pas de jeu de données labellisé. BERT est pré-entraîné sur un grand jeu de données constitué de textes des pages Wikipédia en anglais ainsi qu'un ensemble de livres.

Le prétraitement :

Le prétraitement est la première étape du modèle de BERT. Elle consiste à supprimer le bruit de notre jeu de données. BERT nettoie l'ensemble des données lors du prétraitement, et supprime les doublons.

On effectue une préparation des données au travers de :

- La tokenisation des mots et ajout de tokens de début et de fin de phrase
- D'un marqueur ajouté à chaque phrase pour les distinguer
- De l'ajout d'un marqueur de position à chaque token (mots) pour indiquer sa position

L'encodage

L'apprentissage automatique ne fonctionne pas avec le texte mais fonctionne avec des nombres. Nous devons donc convertir le texte en nombres réels. C'est donc le processus d'encodage. BERT convertit une phrase donnée en un vecteur d'intégration.

Le modèle de BERT :

Nous avons besoin de télécharger deux modèles présents sur TensorFlow Hub, un pour effectuer le prétraitement et l'autre pour l'encodage.

Ensuite, il nous faut créer un modèle à l'aide de TensorFlow. Nous avons le choix entre 2 modèles, le modèle séquentiel et le modèle fonctionnel.

Dans le modèle séquentiel, les couches sont construites les unes sur les autres, couche par couche, avec une seule entrée et une seule sortie.

Les modèles fonctionnels sont plus robustes et flexibles. Ils ne créent pas de couches dans un ordre séquentiel. Dans ce modèle, nous avons plusieurs entrées et sorties.

Dans notre cas, nous utilisons l'approche fonctionnelle pour construire notre modèle.

Initialisation des couches BERT :

Pour l'initiation des couches de BERT, nous créons une couche d'entrée. Une fonction convertit ensuite le texte prétraité en **vecteurs d'incorporation**. Ce sera la sortie de cette couche. Les sorties (outputs) seront par la suite introduites dans les couches du réseau de neurones.

Initialisation des couches du réseau de neurones :

Le réseau de neurones comporte deux couches, la couche "Dropout" et la couche "Dense".

Couche dropout :

Cette couche est utilisée pour empêcher l'overfitting du modèle. L'overfitting se produit lorsqu'un modèle apprend parfaitement à partir des données d'entraînement mais fonctionne mal lors des tests. Nous utiliserons 0.1% des neurones pour gérer l'overfitting.

Couche dense :

Il n'y a qu'un seul neurone. Nous initialisons la fonction d'activation qui est utilisée lorsque nous avons des valeurs de sorties comprises entre 0 et 1.

La compilation du modèle s'effectue sur plusieurs hyperparamètres. Après cette compilation, nous pourrons intégrer le modèle à notre ensemble de données.

Lors de l'ajustement du modèle, il apprend à partir des échantillons de données d'apprentissage et identifie des modèles dans l'ensemble de données de formation et obtient des connaissances.

Evaluation du modèle :

Pour évaluer le modèle, nous avons besoin d'utiliser le modèle pour classer les échantillons de données dans l'ensemble de données de test. Le résultat doit être dans un tableau à une dimension. Ce modèle sera utilisé pour faire une prédiction unique à l'aide de texte d'entrée. Le but de la prédiction c'est de tester nos données sur le modèle que nous avons entraîné et pré-entraîné.

Hyperparamètres pour la compilation du modèle :

L'algorithme de BERT prend en compte plusieurs paramètres lors de la compilation du modèle avec la commande modele.comile() :

- Optimizer : ce paramètre est utilisé pour améliorer les performances du modèle dans l'optique de réduire les erreurs de biais que produit le modèle.
- Loss Function : ce paramètre est utilisé pour calculer l'erreur pendant l'apprentissage du modèle.
- Metrics : ce paramètre est utilisé pour vérifier les performances du modèle, on peut appeler plusieurs métriques de prédiction.

Hyperparamètres pour l’apprentissage du modèle :

L’algorithme de BERT prend en compte plusieurs paramètres lors de l’apprentissage du modèle avec la commande modele.fit() :

- x : Données d’entrées (tableau numpy, un tenseur…)
- y : Données cibles
- batch_size : Nombre d'échantillons par mise à jour du gradient. Peut être un nombre ou None.
- epochs : Nombre d'époques pour former le modèle. Il s’agit d’une itération sur l'ensemble x et y les données fournies. Doit être un entier.
- validation_split : Pourcentage du set de base à utiliser comme jeu de validation.
- validation_data : Données pour la validation du modèle. Doit comprendre (x_valid, y_valid)

III.BERT sur nos données

Ce qui différencie nos modèles effectués au premier semestre de l'algorithme de BERT, est d'une part que non seulement BERT fait intervenir le sens de la phrase où est situé le mot cible, ce qui n'était pas le cas avec les algorithmes de classifications du semestre 1 qui prenaient en compte uniquement le mot cible d'une phrase.

Et d'autre part que BERT est déjà pré-entraîné sur une multitude de données réelles, donc il est acclimaté à de nombreuses observations ciblées, ce qui n'était pas non plus le cas au semestre 1, car nous entraînons nos modèles et les testons sur les mêmes données.

C'est ainsi que l'algorithme de BERT est d'autant plus utile pour pouvoir répondre à notre problématique de départ que les algorithmes de classifications comme le K-means ou encore le SVM.

Au niveau des hyper paramètres choisis pour l'exécution de ce modèle, on s'est tourné vers des paramètres assez simple pour commencer, puis avec perspective de les changer pour pouvoir améliorer les scores obtenus :

- Pour la compilation du modèle :
    - optimizeur='adam' -
    - loss='binary_crossentropy' - définition de la fonction de perte avec binary_crossentropy car la réponse est binaire : 0 ou 1.
    - metrics : accuracy, precision, recall - métriques définies pour prédire les scores du modèle.

- Pour l'entraînement du modèle :
    - epochs=50 - Nous avons choisis de partir sur 50 epochs pour maximiser les chances que le modèle s'améliore.

L'objectif de cet algorithme appliqué à nos données sera donc de lire un post reddit ou twitter et de pouvoir comprendre s'il s'agit d'un comportement suicidaire ou non avec un modèle qui a été entraîné sur nos données wikipédia, qui comme dit précédemment ce sont des données réelles qui regroupent des personnes qui se sont suicidés.

En sortie de l'algorithme, nous nous attendons à avoir des résultats moyennement bon, car pour obtenir des résultats convenable il aurait fallu que les tweets et post reddit soit écrit à la 3ème personne du singulier comme les corpus de données wikipédia, et non pas de manière inclusive avec un 'Je'.

CONCLUSION

Pour conclure, nous avons pu appliquer quatre algorithmes pour classer nos données. Au premier semestre nous avons présenté le SVM et le Random Forest qui se sont avérés avoir de bons résultats, mais aussi le Kmeans qui avait de mauvais résultats dû au fait que ce soit une classification non supervisée. Cela nous a amené à découvrir le Deep Learning pour le second semestre avec l'algorithme de BERT.

SOURCE

K-MEANS :
https://fr.wikipedia.org/wiki/K-moyennes
https://mrmint.fr/algorithme-k-means
https://delladata.fr/kmeans/

SVM :
https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support
https://www.ibm.com/docs/fr/spss-modeler/SaaS?topic=models-how-svm-works
https://larevueia.fr/support-vector-machines-svm/
https://datascientest.com/svm-machine-learning
https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501879-machine-a-vecteurs-de-support-svm-definition-et-cas-d-usage/
https://dataanalyticspost.com/Lexique/svm/

RANDOM FOREST :
https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501905-random-forest-ou-foret-aleatoire-definition-et-cas-d-usage/
https://fr.wikipedia.org/wiki/For%C3%AAt_d%27arbres_d%C3%A9cisionnels
https://blog.ysance.com/algorithme-n2-comprendre-comment-fonctionne-un-random-forest-en-5-min
https://dataanalyticspost.com/Lexique/random-forest/
https://larevueia.fr/random-forest/
https://www.lemagit.fr/conseil/Les-differences-entre-arbre-de-decision-Random-Forest-et-Gradient-Boosting

BERT
https://www.kaggle.com/general/100580
How to Build a Text Classification Model using BERT and Tensorflow | Engineering Education (EngEd) Program | Section
https://ledatascientist.com/a-la-decouverte-de-bert/