Alexis

PLESSIER

27 Novembre 2021

TP Intelligence Artificielle

Rendu

M1 Informatique

Université Clermont-Auvergne

I – Description du problème

J'ai décidé d'orienter mon projet vers un projet de Machine Learning, j'ai donc eux besoin d'avoir accès à un DataSet qui soit à la fois complet, utile et compréhensible. Après avoir épier plusieurs options j'ai choisi de travailler à partir du site Kaggle.com car il possède de nombreux DataSet ainsi qu'un NoteBook intégré. Nous avions également travaillé dessus l'année précédente, je ne serai donc pas dépaysé. Quand au choix de ma problématique (et donc de mon DataSet) j'ai choisi « Heart Failure Prediction Dataset ». Les maladies cardio-vasculaires sont la première cause de mortalité dans le Monde et fais près de 18 millions de victimes chaque année soit 31% des décès. Un tier des maladies cardio-vasculaires font des victimes avant 70 ans. Notre modèle de prédiction aura pour but d'essayer de prédire si un patient possède un fort risque de présenter une de ces maladies selon les indicateurs suivants :

- Son Âge
- Son Sexe
- Son type de douleur Thoracique
- Sa pression sanguine au repos
- Son taux de cholestérol
- Son taux de glycémie à jeun
- Ses résultats d'électrocardiogramme au repos
- Sa fréquence cardiaque maximale atteinte
- S'il possède une angine de poitrine après un effort
- La valeur du segment ST (électrocardiogramme)
- La pente du segment ST après un effort
- Si la personne possède une maladie cardio-vasculaire (pour l'ensemble d'entrainement de la machine)

J'ai donc décidé de mettre en place ce projet sous le langage Python avec les bibliothèques

II – Mise en place du projet

1) Premier résultat

Pour ce premier test, j'ai décidé de prendre ma base de données brut, de modifier les variables qualitatives de type String en variables entières de type Entier, puis d'entrainer mon modèle avec un Classifieur de K voisins plus proches. A l'arrivée, mon modèle donnait un résultat de bonnes prédictions de 74% sur mon ensemble de test, ce qui était loin de mes espérances. Ce premier test avait pour but de mieux me représenter le travail que j'avais à faire de partir sur un résultat de base à améliorer.

2) Pré-Processing

Dans la démarche d'améliorer mon modèle j'ai choisi de m'informer dans les bonnes pratiques à avoir pour obtenir un résultat optimal et l'étape indispensable est d'effectuer une opération de Pré-Processing sur les données, c'est-à-dire, de préparer les données pour qu'elles soient mieux exploitable. Tout d'abord, on vérifie qu'il n'y a pas de données manquantes qui pourrait fausser nos

résultats (par chance ce n'est pas le cas de ce DataSet). Ensuite, on identifie les variables qualitatives et quantitatives pour les mettre à l'échelle. Il existe deux procédés très connus :

- MinMax : Cette méthode va définir la plus petite valeur de la colonne comme 0 et la plus grande comme 1 pour établir un rapport sur toutes les données présente.
- Standardiser : Cette méthode va définir un rapport tel qu'il existe autant de valeur en dessous de la moyenne des données qu'au-dessus.

Après plusieurs essais, il s'est avéré que la Standardisation donner de meilleurs résultats sur cet échantillon de données.

Enfin, on définit nos deux ensembles. Le premier, **X**, représente nos **features** qui seront les indicateurs sur lesquels notre modèle va s'entrainer (tout le tableau sauf *HeartDisease*). Le second, **y**, qui est notre **target** et donc l'indicateur qui prédit si oui ou non la personne est susceptible de présenter une maladie cardio-vasculaire (juste la colonne *HeartDisease*).

3) Entrainement du modèle

Une fois nos TrainSet et nos TestSet définies, nous devons choisir un modèle sur lequel l'entrainer, et comme nous essayons de prédire une catégorie (A un risque ou non), nous allons devoir trouver le meilleur classifieur disponible. Il existe des classifieurs plus adapté à certains DataSet et nous pourrions voir lequel correspond le mieux à notre cas mais en réalité, et ce qui se fait tout naturel sur la plupart des projets, c'est que je vais en essayer un maximum et prendre celui qui me retourne le résultat le plus pertinent. Je vais créer un tableau avec tous mes classifieurs.

Je vais donc pouvoir écrire la fonction qui prendra en argument le classifieur candidat et retournera sa pertinence par rapport à mon DataSet. Pour ceci je vais lier mon classifieur avec mes TrainSet pour qu'il s'entraine mais en utilisant une technique particulière : **la cross-validation**. La cross validation va permettre à mon TrainSet d'être divisé en n parties (*ici 4*), afin d'entrainer le modèle sur n-1 parties sur n du TrainSet et de le valider sur la dernière partie et d'obtenir un score. La cross validation va faire ça sur chaque n fois en changeant à chaque fois la partie de validation pour enfin prendre la moyenne de chaque score.

Cette notion de cross-validation va nous permettre de vérifier un phénomène important pour l'optimisation de notre modèle : **La courbe d'apprentissage**. La courbe d'apprentissage prend en compte le nombre d'entrées et le score obtenu avec ce nombre d'entrées précis. Le plus intéressant est de comparer les courbes du TrainSet et du ValidationSet, 2 cas particuliers se distinguent alors :

- Underfitting (sous-apprentissage) : Il se caractérise par un mauvais apprentissage et donc une mauvaise validation (Score de TrainSet et ValidationSet bas). L'underfitting peut être causé par l'utilisation du mauvais modèle ou encore une non-relation entre les indicateurs et la prédiction à faire.
- Overfitting (sur-apprentissage) : Il se caractérise par un manque de généralisation du TrainSet (Le TrainSet est linéaire et souvent à son maximum, il a tout bien

appris, peut-être même trop bien appris …). L'overfitting peut être causé par sa trop grande capacité à capturer les informations.

4) Optimisation du modèle

Après avoir obtenu les rapports de chaque classifieur pour mon DataSet et avoir trié les courbes d'apprentissages intéressantes, je peux en choisir un ou plusieurs. Dans mon cas j'ai réduit mon choix à deux classifieurs qui sont **AdaBoost** et **SVM**. Chaque classifieur repose sur sa propre méthode et possède donc des *hypers paramètres* distincts qui ont été exécuter avec leur valeur par défaut jusqu'ici. Nous allons désormais essayer d'améliorer la précision de ces classifieurs en « jouant » sur ces paramètres grâce à une fonction : **GridSearchCV**. Cette fonction est simple, nous allons lui donner en argument un classifieur, des valeurs pour ses *hypers paramètres*, la variable à maximiser (ici nous avons choisi **accuracy** car la précision nous intéresse le plus) et le nombre de découpage pour notre cross-validation. La fonction va tester le résultat de accuracy pour chaque combinaison de chaque paramètre sur chaque partie de chaque cross-validation pour nous donner les paramètres qui fonctionnent le mieux. Une fois ceci fait, nous avons notre modèle.

III – Conclusion

Pour conclure, après toutes ces étapes, mon modèle a réussi à prédire 91,3% de bonne réponse pour la matrice de confusion suivante :

N = 184		Classe Prédite	
		0	1
Classe Réelle	0	**67**	**6**
	1	**10**	**101**

Cette précision est, je trouve, un très bon score et seront totalement acceptable sur plein d'autres DataSet. Cependant, nous avons ici affaire au domaine de la santé et plus précisément aux maladies cardio-vasculaires, nous ne pouvons pas nous permettre de passer à coté d'une maladie sur 10 comme nous ne pouvons pas nous permettre de détecter à tort une maladie. Chaque résultat doit être considéré en fonction du domaine qu'il touche et de l'importance qu'il lui est donné.
Je pense qu'il est possible d'améliorer la précision de ce modèle avec plus de *features* comme le poids, l'IMC, ou d'autre indicateurs ainsi que plus de données (nombre de patients). Je retiendrai également l'importance du pré-Processing et de la dataVizalisation qui est propre à chaque problématique et indispensable pour obtenir le résultat le plus convaincant.

IV – Annexes

(Notebook du code)