École Supérieure d'Ingénieurs Léonard de Vinci, La Défense
Datascience et IA, S6 2021

Classification Challenge

Flavien DESEURE–CHARRON

19/04/2021

1 Introduction

L'objectif de ce devoir est de concevoir l'algorithme des k plus proches voisins et de l'utiliser sur un dataset anonymisé afin d'obtenir la meilleure précision possible.
Un second dataset est également à notre disposition pour évaluer le modèle sur des données inconnues. Il pourrait aussi servir à entraîner le modèle avec plus d'exemples.

2 Analyse des données

J'ai commencé ce devoir par une analyse du dataset fournit dans le but de le comprendre du mieux possible.

2.1 Analyse de forme

- **Variables prédictives et cible :**
  On a six variables prédictives quantitatives continues et une variable cible qualitative nominale.
  On est donc dans un problème supervisé de classification.
- **Taille du dataset :**
  Il contient 803 lignes, c'est donc un petit dataset. Il faudra privilégier des solutions d'évaluation du modèle adaptées.
- **Analyse des valeurs manquantes :**
  Aucune valeur manquante.

2.2 Analyse de fond

- **Visualisation de la variable cible :**
  Les classes C et D représentent à elles seules 56% des valeurs. En revanche, la classe E est très peu présente (environ 6% des données).

- **Signification des variables:** Aucune des variables n'est centrée.
  La variable 1 a une grande plage de valeur.
  Les variables 2 et 4 ont une plage de valeur moyenne.
  Les variables 3, 5 et 6 ont une plage de valeur faible.

  Normaliser les données est donc une solution à envisager.

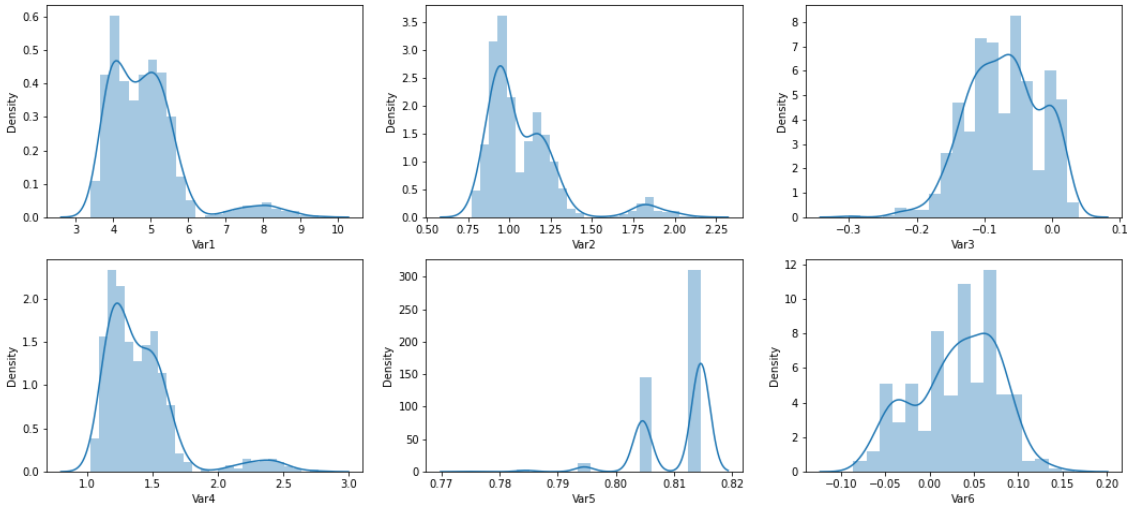

Figure 1: Histogramme des variables

- **Relation variables prédictives/cibles:**
  Les variables 3 et 6 se ressemblent beaucoup et permettent de séparer les classes B, D et le groupe ACE.
  De même les variables 1, 2 et 4 sont très proches et permettent de séparer les classes E, C, D et le groupe AB.
  Enfin, la variable 5 ne permet pas la séparation de quelconques classes.

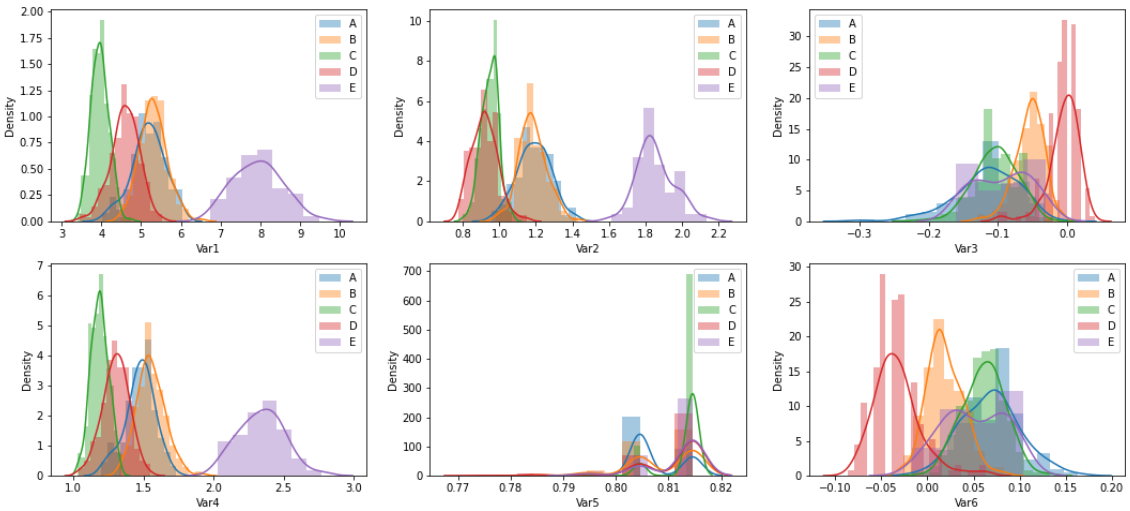

Figure 2: Histogramme des variables en fonction des classes à prédire

- **Relation Variables/Variables**
  Les variables 3 et 6 sont très corrélées négativement et les variables 1, 2 et 4 le sont positivement, ce qui confirme nos observations précédentes.

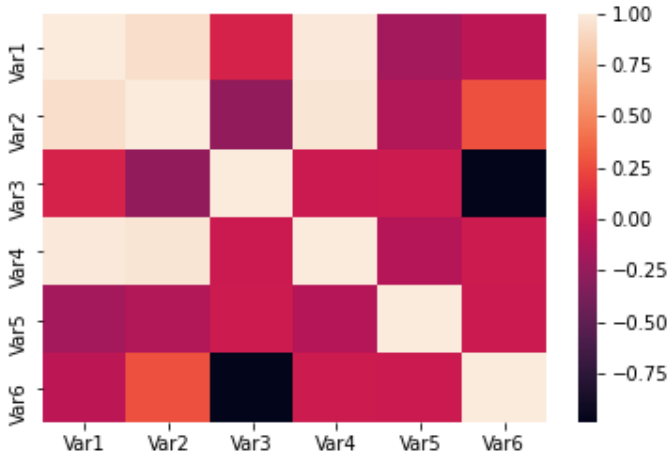

Figure 3: Heat map de corrélation entre les variables

On peut vérifier cela grâce aux graphiques des corrélations entre variables.

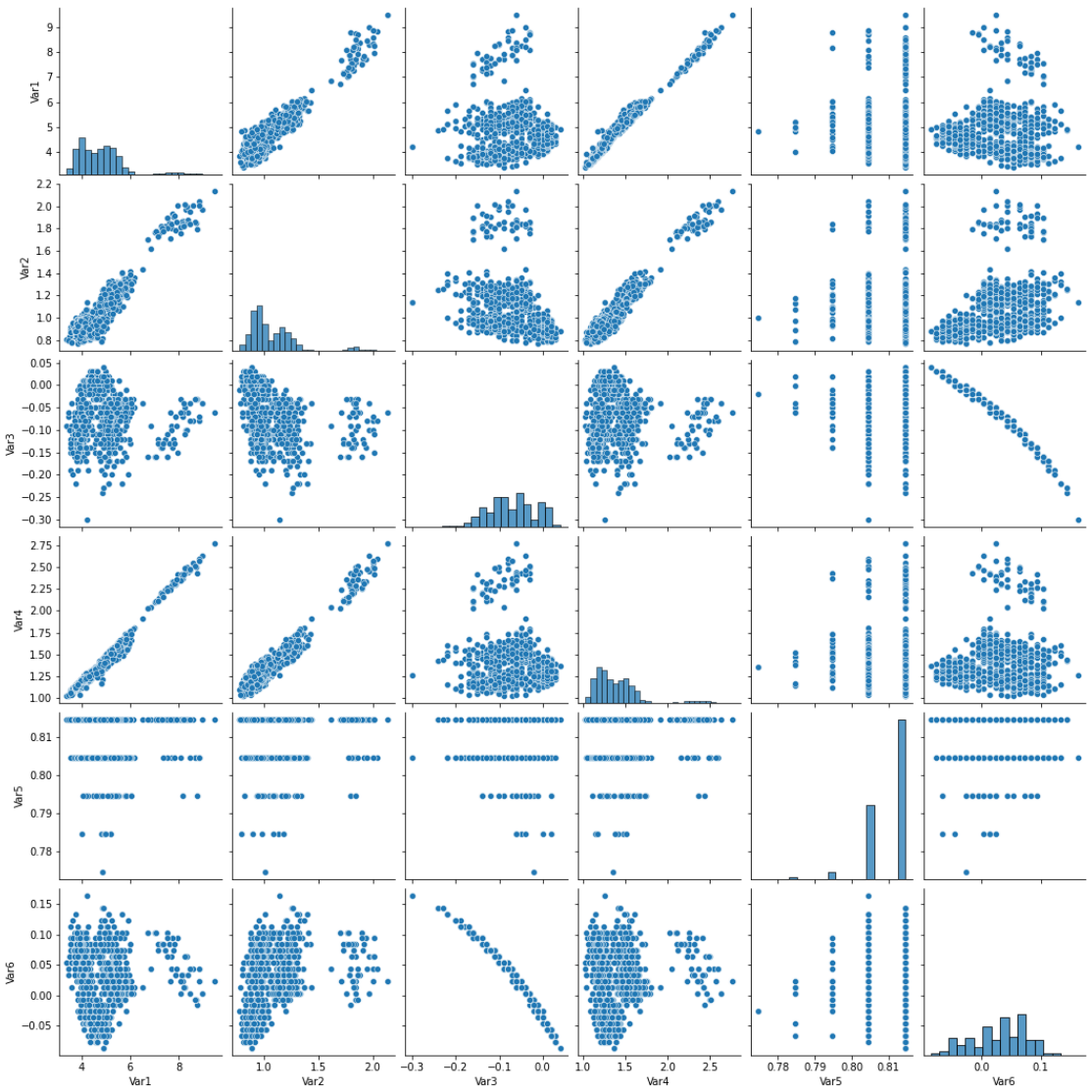

Figure 4: Diagrammes des corrélations entre les variables

Conclusion:
Les variables 1, 2 et 4 sont corrélées positivement entre elles et les variables 3 et 6 le sont négativement.
Comme notre objectif est de trouver la plus grande précision possible, les intégrer ne peut qu'améliorer les performances du modèle.
La variable 5 n'a l'air d'avoir aucun intérêt pour notre modèle (à vérifier par la suite).
On privilégiera donc les variables 1,2,3,4 et 6.

3 Pré-traitement des données

Le pré-traitement des données est une étape importante dans un projet de machine learning. Cette partie consiste à préparer au mieux le dataset qui sera fourni au modèle en utilisant les informations collectées lors de la phase d'analyse des données.
Elle est composée de nombreuses étapes, en voici les principales :

1. **Encodage des données**
   Cette partie consiste à transformer des données qualitatives en données quantitatives. Ici, cela n'est pas nécessaire car toutes nos données sont quantitatives.

2. **Normalisation des données**
   Il existe de nombreuses façons de normaliser ses données. J'en ai testé plusieurs:

   - **MinMax :**

     $$\frac{X - X_{min}}{X_{max} - X_{min}}$$

     Toutes les données sont comprises entre 0 et 1.

   - **Standardisation :**

     $$X_{scaled} = \frac{X - \mu_X}{\sigma_X}$$

     On centre et réduit les données.

   - **Standardisation robuste :**

     $$X_{scaled} = \frac{X - mediane}{IQR}$$

     Cette méthode permet d'avoir une échelle peu sensible aux outliers grâce à l'utilisation de la médiane et de l'écart interquartile.

3. **Imputation**
   On remplace les valeurs manquantes selon des stratégies définies en avance.
   Dans notre cas, ce n'est pas nécessaire.

4. **Sélection de variables**
   On sélectionne les meilleurs variables pour notre modèle.
   Ici, l'analyse des données nous a suggéré de choisir les variables 1, 2, 3, 4 et 6.

5. **Extraction**
   Cela consiste à réduire le nombre de caractéristiques en créant des représentations de données de dimension inférieure et plus puissantes à l'aide de techniques, telles que l'ACP, l'extraction de représentations vectorielles continues et le hachage.
   J'ai effectué une ACP, mais cela n'a pas apporté d'informations supplémentaires utiles au modèle (les performances ne sont pas meilleures).

4 Modélisation

J'ai pu tester de nombreuses combinaisons en m'appuyant sur l'analyse de données réalisée précédemment.
Pour évaluer mon modèle, j'ai choisi d'utiliser les méthodes suivantes :

- **la validation non-croisée**, qui consiste à diviser en trois le dataset, une partie pour l'entraînement (train) du modèle, une partie pour l'optimisation de ses hyper-paramètres (validation) et une partie pour évaluer le modèle (test).
- **la validation croisée à k blocs**, qui consiste à découper le dataset en K sous-ensemble puis prendre un des K sous-ensemble comme dataset de validation et les K-1 restants comme dataset d'entraînement. On répète l'opération sur toutes les combinaisons possibles. On obtient K mesures de performance dont la moyenne représente la performance de l'algorithme.

Sur les 1604 valeurs fournies (en comptant le pre-test), j'ai réalisé deux configurations :

- **Première configuration avec validation :**
  L'objectif de cette première configuration était de faire varier les hyperparamètres du modèle.
  - train : 37.5%
  - validation : 12.5%
  - test : 50% (preTest)
- **Deuxième configuration sans validation :**
  L'objectif de cette deuxième configuration était d'utiliser les informations trouvées avec la première configuration sur un dataset d'entraînement plus large.
  - train : 75%
  - test : 25%

4.1 Première configuration

Afin de gagner du temps, pour chaque paramètre que je fais varier, j'optimise aussi le nombre de voisins k.

4.1.1 Suppression d'une variable

J'ai tout d'abord mesuré l'influence de la suppression de la cinquième variable comme suggéré plus haut.
J'ai obtenu exactement les mêmes performances pour k = 3:
Validation non croisée (données de validation) : 90,55% de précision
Validation croisée : 89,25% de précision

Cela signifie qu'elle n'a aucune influence positive sur le modèle. J'ai donc fait le choix de ne pas la sélectionner.
Ce choix s'est avéré correct. En effet, après normalisation et choix de la distance, j'obtiens 2% d'amélioration en enlevant cette variable (92,04% contre 90,04% avec la validation croisée).

4.1.2 Modification de la distance

La distance utilisée a une grande influence sur l'algorithme. J'ai pu en tester quatre.

- Distance de Manhattan
  $\forall\ (x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n \times \mathbb{R}^n$

  $$d = \sum_{i=1}^{n} |x_i - y_i|$$

  Performances avec k = 3 :
  Validation non croisée (données de validation) : 90% de précision
  Validation croisée : 89,56% de précision

- Distance euclidienne
  $\forall\ (x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n \times \mathbb{R}^n$

  $$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

  Performances avec k = 3 :
  Validation non croisée (données de validation) : 90,55% de précision
  Validation croisée : 89,25% de précision

- Distance de Minkowski
  $\forall\ (x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n \times \mathbb{R}^n,\ p \geq 1$

  $$d = \sqrt[p]{\sum_{i=1}^{n} (x_i - y_i)^p}$$

  Performances avec k = 3 :
  Validation non croisée (données de validation) : 90,54% de précision
  Validation croisée : 89,56% de précision

- Cosinus distance
  $\forall u, v \in \mathbb{R}^n \times \mathbb{R}^n$

  $$d = \frac{u \cdot v}{\|u\|\|v\|}$$

  Performances avec k = 3 :
  Validation non croisée (données de validation) : 4% de précision
  Validation croisée : 2,39% de précision

J'en ai donc conclu que la distance la plus adaptée pour notre problème ici est la distance de Manhattan.
En effet, elle a les meilleures performances et le temps de calcul le plus court (par rapport à la distance de Minkowski).

4.1.3 Normalisation des données

- Sans normalisation :
  Performances avec k = 3 :
  Validation non croisée (données de validation) : 90,04% de précision
  Validation croisée : 90,04% de précision

- MinMax : Performances avec k = 5 :
  Validation non croisée (données de validation) : 92,04% de précision
  Validation croisée : 92,04% de précision

- Standardisation Performances avec k = 5 :
  Validation non croisée (données de validation) : 92,04% de précision
  Validation croisée : 92,04% de précision

- Standardisation robuste Performances avec k = 5 :
  Validation non croisée (données de validation) : 92,04% de précision
  Validation croisée : 92,04% de précision

J'en ai donc conclu que standardiser les données est la méthode la plus adaptée pour notre problème.

4.1.4 KNN pondéré en fonction de la distance

Cette méthode consiste à pondérer les points par l'inverse de leur distance.
Dans ce cas, les voisins les plus proches d'un point inconnu auront une plus grande influence que les voisins plus éloignés.
Performances avec k = 1 :
Validation non croisée (données de validation) : 92,04% de précision
Validation croisée : 87,86% de précision
Les performances étant trop faibles, je n'ai pas gardé cette alternative.

4.1.5 Remarques

- J'ai pu analyser la matrice de confusion pour chaque résultat trouvé et il semblerait que mon modèle a dû mal à décerner les classes A et B. On remarque effectivement que ses deux classes ne sont pas facilement séparables (voir figure 5).

- J'ai réalisé une ACP sur mes données, cependant, cela n'a pas permis d'améliorer les performances du knn.

4.1.6 Conclusion

Après analyse de l'ensemble de ces résultats, la version optimale trouvée comprend la suppression de la cinquième variable, la standardisation des données, l'utilisation de la distance euclidienne et un nombre de voisins égal à 5.

Cette combinaison me donne les taux de précision suivants :
Validation non croisée (données de validation) : 92,04% de précision
Validation non croisée (données de test) : 83,19% de précision
Validation croisée : 92,04% de précision

4.2 Deuxième configuration

Afin de vérifier si mon modèle a suffisamment de données d'entraînement, j'ai tracé la courbe d'apprentissage :

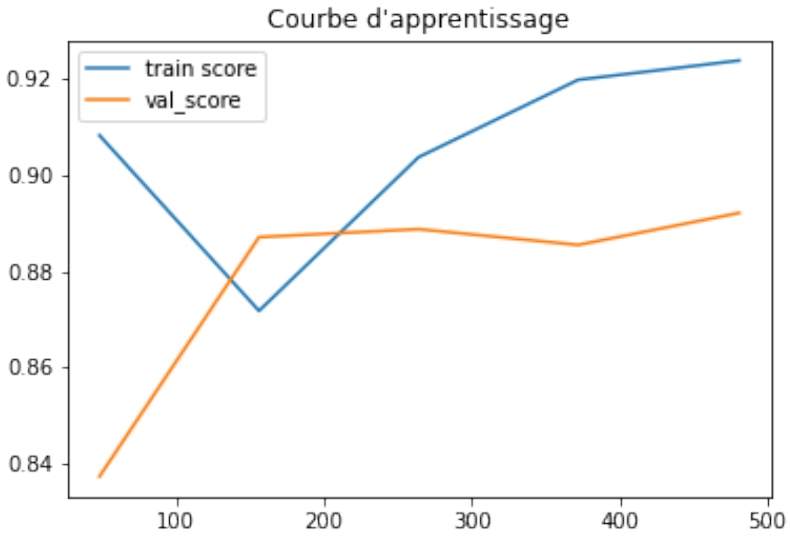

Figure 5: Histogramme des variables en fonction des classes à prédire

De cette courbe, on peut en déduire que rajouter des données d'entrainement nous permettrait d'avoir de meilleurs performances (pas d'overfitting).

Avec la première configuration mais en doublant les données d'entraînement (1204 valeurs contre 602) on obtient les performances suivantes avec k = 5:
Validation non croisée : 90,3% de précision
Validation croisée :

- **Moyenne :** 90,3%
- **Écart type :** 0,16%

- **Minimum:** 90,05%
- **Maximum :** 90,55%

Finalement, j'ai remarqué que la distance euclidienne s'avérait meilleur que la distance de Manhattan dans cette configuration.

D'après les indications données par M. Rodrigues, le dataset final se rapprocherait du dataset preTest. Une comparaison des proportions avec le dataset final donne un taux de différence de 13,14%.
Cet indicateur n'a aucune valeur théorique, mais il permet, tout de même, d'avoir un aperçu du comportement du modèle sur des nouvelles données.

5 Conclusion

J'ai donc choisi la deuxième configuration pour mon modèle qui présente de bonne performances sur des données inconnues avec une grande stabilité.