Projet Méthodologie

Par Assia BOUZNAD

Master sciences du langage : Langue et informatique
Enseignement : Méthodologie de la recherche
Enseignant : Laurence DEVILLERS, Nour BENCHAABENE
2023-2024

Table des matières

I. Introduction

Ce projet a pour objectif d'élaborer un programme permettant de classifier automatiquement des données. Le programme devra être capable de prédire des étiquettes à partir de données d'entraînement. Nous cherchons à adopter une méthodologie efficace pour optimiser les résultats de prédiction. Le programme est écrit en Python, et les données d'entraînement ainsi que de test proviennent du site « kaggle.com ». Pour ce faire, nous avons décidé de travailler sur trois jeux de données distincts.
Le premier jeu de données (corpus 1) servira à prédire si une personne est fumeuse ou non en fonction de l'état de santé de sa cavité buccale. Les données de ce jeu sont constituées de chiffres et de lettres simples indiquant si un individu possède un attribut spécifique ou non (par exemple, 'Y' pour 'yes' et 'N' pour 'no').
Le second jeu de données (corpus 2) sera utilisé pour prédire la réussite des étudiants en fonction de leur consommation d'alcool. Les données de ce corpus sont exclusivement numériques.
Enfin, le troisième jeu de données (corpus 3) aura pour objectif de prédire le genre musical en fonction de paramètres musicaux, tels que le tempo, la valence, etc. Ces données sont majoritairement numériques, à l'exception des étiquettes qui contiennent les noms des genres musicaux.

Dans un premier temps, nous établirons un état de l'art pour chaque jeu de données afin de recenser les travaux effectués précédemment. Ensuite, nous présenterons les jeux de données. Enfin, après avoir détaillé la méthodologie adoptée pour ce projet, nous exposerons les résultats obtenus.

II. Etat De L'art

La méthodologie de recherche en informatique permet d'acquérir un savoir-faire pour manipuler ou visualiser des données grâce à des programmes informatiques élaborés à partir des langages/algorithmes. Dans ce projet, nous avons utilisé le langage python, qui est un langage de programmation. Il permet de faire de la programmation orientée objet (python, 2001-2004). L'apprentissage machine permet de résoudre des difficultés rencontrées dans de nombreux champs comme la médecine ou l'automobile (JannyS. ed, 2022).
Comme expliqué précédemment, ce projet présente trois datasets différents.
Le dataset 'smoking' (Corpus 1) a été utilisé lors d'un concours organisé sur le site «Kaggle». Le participant Ida Bagus Dwiweka Naratama (2023) a utilisé python pour classifier les données et prédire si une personne fume en fonction de son état de santé. Après avoir nettoyé les données, il a utilisé le classifieur «LGBM» et a obtenu un résultat de 87 % de prédictions correctes.

Ensuite, le second dataset 'alcohol' (corpus 2) a été utilisé lors d'études pour prédire la réussite des étudiants en fonction de leur profil et de leur consommation d'alcool. Les données sont partagées en deux: les résultats obtenus par les élèves en mathématiques et ceux obtenus en langue portugaise. Nous avons seulement utilisé le dataset sur les résultats scolaires en mathématiques; nous nous intéresserons donc uniquement aux résultats obtenus avec ce dataset dans l'article. L'article est intitulé «USING DATA MINING TO PREDICT SECONDARY SCHOOL STUDENT PERFORMANCE» et a été rédigé par Paulo Cortez et Alice Silva. Ils ont utilisé le langage R (R Development Core Team 2006). Ils ont utilisé 5 classifieurs différents; les meilleurs résultats sont présentés sous forme de tableau. Nous nous intéressons aux deux dernières colonnes des tableaux (B et C), qui se rapprochent des données exploitées dans notre code. Les meilleurs résultats obtenus sont de 83,8 % de bonnes prédictions avec le classifieur Naïves Bayes.

Enfin, le dernier dataset 'music' (corpus 3) a été utilisé pour une étude nommée «MACHINE LEARNING APPROACH FOR GENRE PREDICTION ON SPOTIFY TOP RANKING SONGS», réalisée par Kehan Luo. Le dataset n'est pas identique à celui que nous avons utilisé; cependant, il est relativement similaire et présente les mêmes types de données. Il a obtenu un résultat de 46,9 % avec le classifieur SVM.

III. Presentation Du Corpus (Jeux De Données)

Le jeu de donnée ‘smoking’ (corpus 1) occupe un espace de 6.32MB. Le tableau csv est composé de 27 colonnes qui comportent 26 instances et 55972 lignes, représentant le nombre de personnes interrogées sur leur santé et leur consommation de cigarette. Dans notre programme, nous avons décidé de sélectionner les instances concernant la santé buccale de chaque individu : 'dental caries’ pour les caries, ’tartar' pour la quantité de tartre présent sur les dents, ‘oral' pour l’examen buccal, et ‘Gtp' pour le catalyseur ‘γ-GTP’. Cette dernière instance ne concerne pas l’état buccal de l’individu, cependant nous l’avons ajoutée pour obtenir une meilleure précision. Nous avons 1 instance pour 2 classes : l’instance ‘smoking’, qui renvoie 2 étiquettes : la valeur 1 si l’individu est un fumeur et la valeur 0 s’il ne l’est pas. Les données ont été divisées en deux : 70 % des données sont consacrées à l’entraînement (train), le reste, soit 30 %, est réservé au test de l’apprentissage de la machine (test). Nous obtenons donc un total de 38984 exemples dans le 'training set' et 16708 dans le 'test set'.

Enfin, le jeu de donnée ‘alcohol’ (corpus 2) occupe un espace de 42.38 kB. Le tableau csv est composé de 33 colonnes qui comportent 32 instances et 396 lignes, représentant le nombre de personnes interrogées sur leurs résultats scolaires et leurs informations personnelles. Dans notre programme, nous avons décidé de sélectionner les instances concernant la consommation d’alcool: ‘Dalc' pour la consommation d’alcool durant la semaine (de 1 pour une consommation faible, à 5 pour une consommation très forte), et ‘Walc' pour la consommation d’alcool durant les weekends (de 1 pour une consommation faible, à 5 pour une consommation très forte). Nous avons 1 instance pour 2 classes: l’instance ‘G3’, qui représente la note en mathématiques obtenue à la fin du 3ème trimestre scolaire. Cela renvoie 2 étiquettes : la valeur positive si l’individu a un bon niveau scolaire, c’est-à-dire des résultats supérieurs ou égaux à 10/20 en maths; et la valeur négative si l’individu a un mauvais niveau scolaire, avec des résultats strictement inférieurs à 10/20 en maths. Les données ont été divisées en deux: 70% des données sont consacrées à l’entraînement (train), le reste, soit 30%, est réservé au test de l’apprentissage de la machine (test). Nous obtenons donc un total de 276 exemples dans le 'training set' et 119 dans le 'test set'.

Ensuite, le jeu de donnée ‘music’ (corpus 3) occupe un espace de 3.75MB. Le tableau csv est composé de 24 colonnes qui comportent 23 instances et 9198 lignes, représentant les musiques de spotify. Dans notre programme, nous avons décidé de sélectionner les instances suivantes: 'tempo', 'acousticness', 'danceability', 'liveness', 'speechiness', 'instrumentalness', 'mode', 'key', et ‘energy’. Elles permettent de décrire chaque musique. Nous avons 1 instance pour 8 classes : l’instance ‘genre’, qui renvoie 8 étiquettes. Cependant, après avoir obtenu les résultats, nous nous sommes rendu compte que certains genres, comme le ‘blues’, par exemple, n’étaient jamais reconnus. Nous avons donc décidé de sélectionner 3 classes, les plus reconnues par le programme: ‘classical', ‘hiphop', et ‘jazz'. Les données ont été divisées en deux: 70% des données sont consacrées à l’entraînement (train) et 30% des données sont réservées au test de l’apprentissage de la machine (test). Nous obtenons donc un total de 2111 exemples dans le 'training set' et 905 dans le 'test set'.

IV. Méthodologie

Tout d’abord, pour les trois code, nous avons décidé d’utiliser la méthode de l’apprentissage supervisé. On dispose de données d’entraînement déjà étiquetées, les sorties sont connues. On va entraîner le modèle à partir des données déjà étiquetées (train) pour qu’il puisse prédire la sortie pour des entrées non étiquetées (test).
Les codes diffèrent au niveau du traitement des données car les datasets sont différents. Cependant, le partage des données pour l’entraînement à 70% et les tests à 30%, ainsi que les algorithmes utilisés restent les mêmes. Nous avons décidé de travailler avec 5 classifieurs pour pouvoir les comparer et déterminer quel classifieur est plus adapté à un certain dataset. Les classifieurs utilisés sont les suivants : le Perceptron, Naives Bayes, SVC (svm), Arbre de décision et KNeighbors. Aussi, nous avons adapté les paramètres des classifieurs pour qu’ils soient plus adaptés.

Pour le premier code ‘smoking’, les caractéristiques utilisées sont : 'dental caries', 'tartar', ‘oral' et ‘Gtp'. Pour extraire ces caractéristiques, nous avons utilisé la méthode 'astype(str)' pour convertir les colonnes en objets de type chaîne de caractères. Les différentes étapes de la chaîne de traitement sont :

- La lecture du fichier "smoking.csv" dans la variable ‘data_smoking’.
- La séparation des caractéristiques et de la variable cible en X et y.
- La division des données en ensembles d'entraînement et de test à l'aide de la méthode ‘train_test_split’.
- L’affichage du nombre d'exemples dans chaque ensemble.
- L'initialisation de l'objet DictVectorizer dans la variable ‘V’.
- La transformation des données d'entraînement en dictionnaires et l’encodage avec DictVectorizer.

Pour le second code ‘alcohol’, les caractéristiques utilisées sont 'Dalc' et ‘Walc'. Pour extraire les données, nous avons utilisé la méthode 'read_csv' de la bibliothèque pandas pour lire le fichier CSV contenant les données et les méthodes 'fit_transform' et 'transform' de la classe 'StandardScaler' de la bibliothèque scikit-learn pour mettre à l'échelle les caractéristiques.
Nous avons suivi les étapes suivantes pour le traitement des données :

- La lecture du fichier CSV contenant les données.
- L’extraction des caractéristiques 'Dalc' et 'Walc'.
- La classification des notes en catégories 'positif' ou 'negatif' en utilisant la fonction 'classify_notes'.
- La mise à l'échelle des caractéristiques en utilisant la classe 'StandardScaler'.
- La division des données en ensembles d'entraînement et de test en utilisant la méthode 'train_test_split'.
- L’affichage du nombre d'exemples dans l'ensemble d'entraînement et l'ensemble de test.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data_alcohol = pd.read_csv("Maths.csv")
X = data_alcohol[['Dalc', 'Walc']]
y = data_alcohol['G3']

def classify_notes(note):
    if note >= 10:
        return 'positif'
    else:
        return 'negatif'

data_alcohol['G3_category'] = data_alcohol['G3'].apply(classify_notes)
y = data_alcohol['G3_category']
```

Ici, la fonction ‘classify_notes’ pour classer les notes en "positif" ou "negatif" en fonction d'une valeur seuil (10 dans ce cas), a permis d’améliorer les résultats car les notes dans l’instance ‘G3’, allant de 0 à 20, étaient trop nombreuses. Diviser les données en 2 a permis de faciliter la classification et de mieux

comprendre l'objectif du code qui était de relier la réussite scolaire à la consommation d'alcool des élèves. Pour le dernier code 'music', les caractéristiques utilisées sont 'tempo', 'acousticness', 'danceability', 'liveness', 'speechiness', 'instrumentalness', 'mode', 'key', 'energy'. Nous avons lu les fichiers CSV comme. Pour les deux codes précédents, nous avons filtré le dataframe du "data_music" pour ne conserver que les lignes dont la valeur de la colonne 'genre' est 'hiphop', 'jazz' ou 'classical'.

Les classifieurs (scikit-learn.org):

Pour le perceptron, nous avons modifié le 'random state' de manière aléatoire entre 1 et 10 pour obtenir les meilleurs résultats

```
from sklearn.linear_model import Perceptron
ppn = Perceptron(eta0=0.1, random_state=6)
```

En fonction du dataset, les renomme state avaient une incidence sur la précision de la perception. Par exemple, pour le code ci-dessus provenant du code 'music', les random state est de 6 car cela a permis d'améliorer les résultats du classifieur alors que pour le code 'alcohol', les random state reste nul, il n'a pas d'incidence sur la précision.

Le classifieur Naive Bayes ne possède pas de paramètres modifiables, nous avons donc fait appel à d'autres variantes du classifieur. Nous avons testé BernoulliNB, MultinomialNB et GaussianNB. Nous avons jonglé entre ces trois possibilités pour obtenir le meilleur résultat.

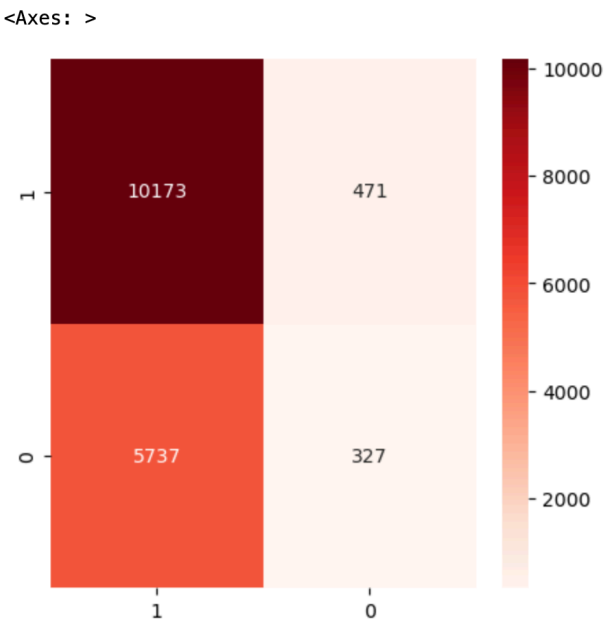
Concernant les classifieurs SVM, plusieurs possibilités de combinaison de paramètres étaient possibles. Dans l'ensemble du projet, seul le paramètre 'kernel', qui permet de mettre les données en relation grâce à différentes fonctions, a été sollicité. Nous avons utilisé les paramètres suivants : 'poly', 'linear', 'rbf' et 'sigmoid'. Le paramètre C, qui augmente la répétition de l'entraînement du classifieur, certes améliore les résultats obtenus, cependant plus le nombre est élevé, plus le classifieur est lent, voire il ne répond plus.

Les arbres de décision ont à disposition des paramètres pour modifier la profondeur des 'feuilles' de l'arbre, cependant, dans notre projet, les paramètres n'ont pas eu une grande incidence sur nos résultats. Enfin, pour les classifieurs KNeighbors, nous avons modifié le paramètre 'metric' en particulier. Parmi les nombreuses options, nous avons travaillé sur 3 valeurs : 'euclidean', 'manhattan' et 'chebyshev' (ou encore 'minkowski' avec p=1 => euclidean, p=2 => manhattan).

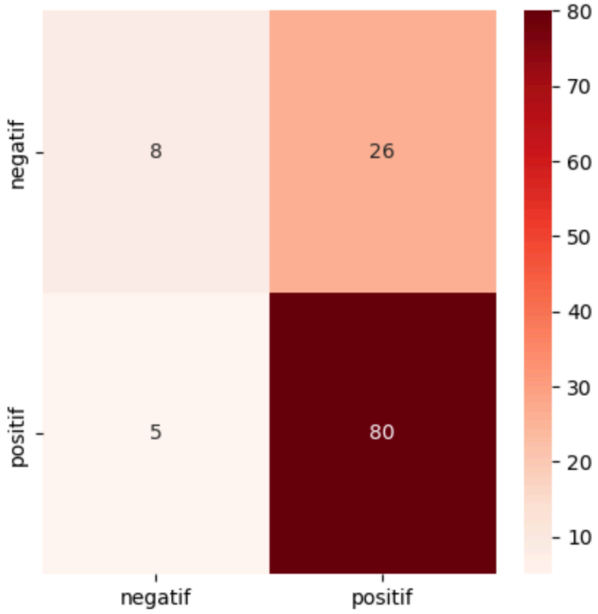
V. Résultats

Classes évalué :

'smoking'



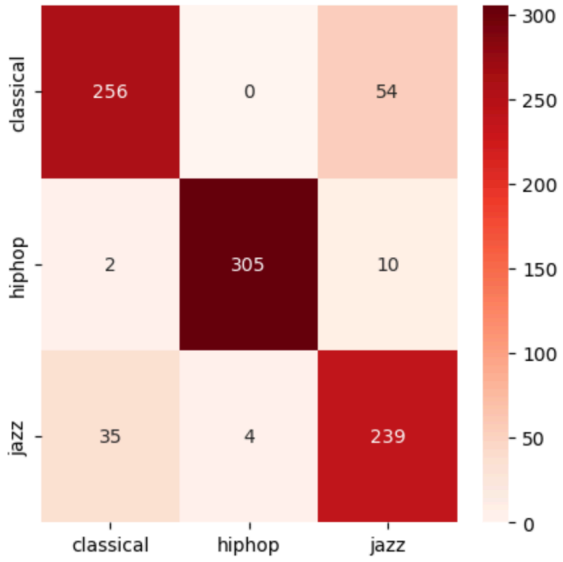
<Axes: >



‘alcohol’

‘music’

<Axes: >



Classifieur :

Le code ‘smoking’ montre que les classifieurs Naives Bayes, SVC et Arbres de decision sont les plus efficaces.

```
]: from sklearn.metrics import accuracy_score

accuracy_perceptron = accuracy_score(y_test, y_pred)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_clf = accuracy_score(y_test, y_pred_clf)
accuracy_knn = accuracy_score(y_test, y_pred_knn)

print('Exactitude pour Perceptron: %.2f' % accuracy_perceptron)
print('Exactitude pour Naive Bayes: %.2f' % accuracy_nb)
print('Exactitude pour SVM: %.2f' % accuracy_svm)
print('Exactitude pour clf: %.2f' % accuracy_clf)
print('Exactitude pour knn: %.2f' % accuracy_knn)
```

```
Exactitude pour Perceptron: 0.67
Exactitude pour Naive Bayes: 0.69
Exactitude pour SVM: 0.69
Exactitude pour clf: 0.69
Exactitude pour knn: 0.67
```

Exactitude :

```
from sklearn.metrics import accuracy_score

accuracy_perceptron = accuracy_score(y_test, y_pred)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_clf = accuracy_score(y_test, y_pred_clf)
accuracy_knn = accuracy_score(y_test, y_pred_knn)

print('Exactitude pour Perceptron: %.2f' % accuracy_perceptron)
print('Exactitude pour Naive Bayes: %.2f' % accuracy_nb)
print('Exactitude pour SVM: %.2f' % accuracy_svm)
print('Exactitude pour clf: %.2f' % accuracy_clf)
print('Exactitude pour knn: %.2f' % accuracy_knn)
```

```
Exactitude pour Perceptron: 0.71
Exactitude pour Naive Bayes: 0.71
Exactitude pour SVM: 0.71
Exactitude pour clf: 0.72
Exactitude pour knn: 0.74
```

Le code ‘alcohol’ montre que le classifieur KNeighbors est plus efficace pour le dataset ‘Maths’. Les résultats des autres classifieurs restent proche.

Le code ‘music’ montre que le classifieur SVM est le plus efficace avec une exactitude de 88%. Pour ce code on remarque que les autres classifieurs ont des résultats relativement proche a l’exception du perceptron qui a un résultat en dessous de 50%.

```
from sklearn.metrics import accuracy_score

accuracy_perceptron = accuracy_score(y_test, y_pred)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_clf = accuracy_score(y_test, y_pred_clf)
accuracy_knn = accuracy_score(y_test, y_pred_knn)

print('Exactitude pour Perceptron: %.2f' % accuracy_perceptron)
print('Exactitude pour Naive Bayes: %.2f' % accuracy_nb)
print('Exactitude pour SVM: %.2f' % accuracy_svm)
print('Exactitude pour clf: %.2f' % accuracy_clf)
print('Exactitude pour SVM: %.2f' % accuracy_knn)
```

```
Exactitude pour Perceptron: 0.47
Exactitude pour Naive Bayes: 0.85
Exactitude pour SVM: 0.88
Exactitude pour clf: 0.86
Exactitude pour SVM: 0.73
```

VI. Conclusion

Pour conclure, les résultats obtenus pour les trois codes peuvent être améliorés. Si on compare nos résultats avec les travaux effectués avant nous, on remarque que l’exactitude atteint les 90 %. On aurait pu utiliser d’autres classifieurs plus adaptés à nos datasets. De plus, nous n’avons pas exploité la totalité des données disponibles dans ce dataset, nous avons sélectionné certaines instances, cela a pu contribuer à l’obtention de résultats plus faibles. Nos codes sont assez génériques, on peut les appliquer sur d’autres datasets, en effet, nos codes peuvent correspondre à d’autres datasets, peu importe la langue, seuls les noms des instances doivent être modifiés.

VII. Bibliographie

Python (2001-2004).Tutoriel. URL: https://docs.python.org/fr/3/tutorial/

Janny S. ed, (2022). *Introduction à l'apprentissage Automatique.* Culture science de l'ingénieur. URL : https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14512/14512-introduction-lapprentissage-automatique-ensps.pdf

Dataset 'music'. Kaggle. URL :

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjl7oa3yMmDAxWbVqQEHY5MDw4QFnoECCQQAQ&url=https%3A%2F%2Fcdr.lib.unc.edu%2Fdownloads%2F8k71nn03b%3Flocale%3Den&usg=AOvVaw3HY7gxYCDLU7JAbUOvt03-&opi=89978449

Dataset 'smoking'. Kaggle. URL :

https://github.com/sowmyamaddali/Body-Signal-of-Smoking#body-signal-of-smoking
https://medium.com/@dwiwekan/binary-prediction-of-smoker-status-using-bio-signals-kaggle-playground-competition-543a13b70248

Dataset 'alcohol'. Kaggle. URL :

http://www3.dsi.uminho.pt/pcortez/student.pdf

Scikit-learn. Tutoriel. URL : https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py