Projet classification
Amina-M'bariqui Ahamada et Soukaina Bouhaid

Table des matières

I. Introduction

II. Présentation des datasets et des objectifs

III. État de l'art

IV. Test des classifieurs et interprétation des résultats

V. Méthodologie

VI. Résultats

VII.Conclusion

Bibliographie

I. Introduction

La classification est une technique polyvalente qui permet de prédire l'appartenance d'un élément dans une catégorie en fonction de ces caractéristiques. Ce projet s'inscrit dans le cadre de cet usage, nos objectifs sont de tester différents algorithmes de classification et de déterminer lequel est le plus efficace, ce qui nous permettra de mieux comprendre leurs fonctionnements. Nous utiliserons les classifieurs Perceptron, Random Forest , Logistic Regression et Support Vector Machine. Puis nous chercherons à améliorer les résultats obtenus en manipulant différents paramètres pour optimiser le résultat de classification ou de durée d'exécution. Nous avons 2 datasets, ayant des données catégorielles et quantitatives.
Premièrement, nous présenterons une analyse exploratoire de nos jeux de données, ensuite nous détaillerons les étapes effectuées pour déterminer le classifieur le plus performant dans la phase de test. Après avoir sélectionné le classifieur le plus efficace, nous tenterons d'améliorer les résultats obtenus avec celui-ci.

II. Présentation des datasets et des objectifs

Le premier jeu de données à notre disposition est le dataset Student Health au format csv. Ce dataset possède 14 colonnes ainsi que 1000 instances par colonne. Ce dataset est composé de variables catégorielles qui sont les suivantes :

- Mood
- Physical_Activity
- Sleep_Quality

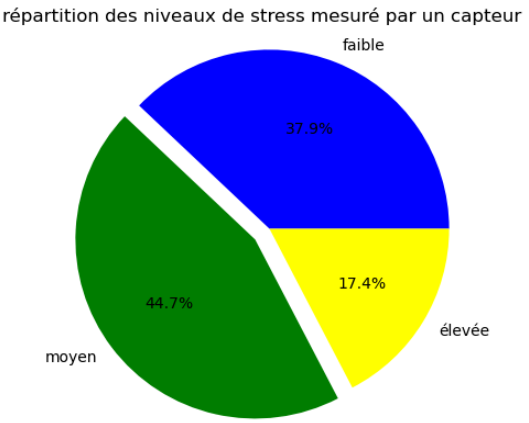
Il possède aussi des variables quantitatives :

- Student_ID
- Age
- Gender
- Heart_Rate
- Blood_Pressure_Systolic
- Blood_Pressure_Diastolic
- Stress_Level_Biosensor
- Stress_Level_Self_Report

Nous avons pour objectif de classer les valeurs de niveau de stress mesuré avec un capteur (Stress_Level_Biosensor), en utilisant comme variables explicatives le rythme cardiaque (Heart_Level) et le niveau de stress que l'étudiant estime avoir (Stress_Level_Self_Report).

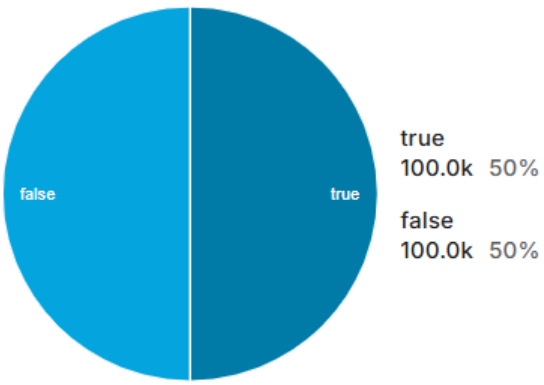
Avant d'effectuer la classification, il serait intéressant de visualiser notre variable cible pour pouvoir mieux interpréter les résultats.

Nous pouvons constater que le niveau “moyen” est surepresenté tandis que le niveau “élevée” est sous-représenté.



Le deuxième dataset que nous avons choisis est Shorts text humor detection au format csv également. Ce dataset a 2 colonnes qui contiennent 200 000 données chacunes. La première colonne est “text” qui possède des textes courts en anglais. La deuxième colonne est “humor” qui contient des valeurs booléens, répartis de manière égal :

Répartition des valeurs true et false dans la colonne “humor”



L’objectif que nous nous sommes fixés avec ce dataset est de réduire le temps d'exécution à moins d’une heure avec les opérations de pré-traitement de texte.

III. État de l'art

- Student health

Dans l'article "Machine Learning-Based Prediction of Mental Well-Being Using Health Behavior Data from University Students" du Bioengineering. Les chercheurs obtiennent les résultats suivant :

Table 1. Model evaluation metrics of machine learning classifiers on the testing data set.

Classifier	Accuracy	Kappa	Sensitivity	Specificity	AUC
Random Forest	0.901	0.801	0.980	0.815	0.966
Random forest + text	0.881	0.759	0.965	0.782	0.951
Adaptive boosting	0.893	0.785	0.951	0.828	0.959
k-nearest neighbor	0.795	0.593	0.711	0.887	0.886
Naïve Bayes	0.702	0.399	0.775	0.621	0.678
Bagging (Bootstrap Aggregation)	0.672	0.349	0.617	0.735	0.677
Recursive partitioning	0.633	0.268	0.607	0.662	0.665
Neural Network	0.615	0.231	0.597	0.633	0.674
Generalized linear model	0.603	0.201	0.673	0.527	0.653

Ils ont tenté de déterminer le niveau de santé mental des étudiants avec les caractéristiques suivantes: the number of sports activities per week, body mass index, grade point average (GPA), sedentary hours, and age. En utilisant un dataset contenant 15 366 instances et 20 features.

IV. Test des classifieurs et interprétations des résultats

Nous avons testé 4 classifieurs : RandomForest Classifier, Perceptron, Logistic Regression et Support Vector Machine afin de déterminer lequel est le plus efficace en matière de classification et de temps d'exécution. Le processus de test se fait avec des données qui n'ont pas été pré-traité dans le but d'observer la performance des classifieurs avec les données brutes.

Avant de tester les classifieurs, il est nécessaire de s'assurer que nos données sont correctement formatés pour pouvoir être gérées par les algorithmes de classification :

Student Health

→ Traitement des données

Le dataset Student Health contient des données catégorielles qui doivent être numérisées pour être traitées par les classifieurs. Pour cela, nous avons utilisé le OneHotEncoder de la bibliothèque scikit-learn. Une fois que nos données catégorielles ont été numérisées, nous les avons joint avec nos données quantitatives avec la fonction hstack disponible dans la bibliothèque Numpy qui permet d'empiler des tableaux.

Notre variable cible Stress_Level_Biosensor a des données quantitatives et continues c'est-à-dire que les données prennent une quelconque valeur entre 1 et 10, mais pour réaliser la classification nous avons besoin de données quantitatives et discrètes. La fonction cut de la bibliothèque pandas permet de segmenter des données en numérique en différents intervalles et d'associer ces intervalles à des labels. Ce qui nous permet de transformer notre variable cible en données quantitatives et discrètes. Ainsi nos catégories sont:

- Les valeurs de [0-4.5[appartiennent à la catégorie "faible".
- Les valeurs de [4.5-8.5[appartiennent à la catégorie "moyen".
- Les valeurs de [8.5-10] appartiennent à la catégorie "élevée".

Nous avons choisi ces intervalles après avoir observé visuellement les valeurs de notre variable cible.
70% des données sont utilisés pour la phase d'apprentissage et 30% sont utilisés pour la phase test.

→ Test des classifieurs

Le test a été réalisé en utilisant Stress_Level_Biosensor comme variable cible, tandis que les autres caractéristiques ont été utilisées comme variable explicative. Voici les résultats obtenus :

	Perceptron	Random Forest	SVM	Logistic Regression
Bonnes classifications	68	68	181	68
Erreurs	232	232	119	232
Accuracy	0.23	0.23	0.60	0.23
Temps d'exécution	0.17 sec	0.53 sec	117 sec	0.60sec

Les classifieurs Perceptron, Random Forest et Logistic Regression ont obtenu des résultats similaires. Cependant, les poids qu'ils ont attribués aux features sont différents, les features ayant les poids les plus importants dans ces classifieurs sont :

Perceptron :

1. Heart_Rate 99.243322
2. Blood_Pressure_Systolic 99.068059
3. Student_ID 98.900000

Random Forest :

1. Stress_Level_Self_Report 0.163754
2. Health_Risk_Level_Low 0.092645
3. Blood_Pressure_Systolic 0.090228

Logistic Regression :

1. Health_Risk_Level_Low 2.174315
2. Health_Risk_Level_High 1.769461
3. Health_Risk_Level_Moderate 0.413063

Bien que les résultats de ces classifieurs soient identiques, le processus utilisé pour réaliser les classifications est différent car ils n'ont pas attribué un poids important au même features.

Le modèle SVM a obtenu de meilleurs résultats que les autres modèles avec un temps d'exécution plus long, ce modèle est donc plus complexe et efficace que les autres classifieurs qui ont été testés. Mais les résultats obtenus sont moins satisfaisants que l'article scientifique. Nous allons donc sélectionner ce classifieur pour poursuivre l'amélioration des résultats et la réduction du temps d'exécution.

Shorts text humor detection

Pour le dataset Shorts text humor, nous avons appliqué les mêmes classifieurs que pour le dataset Student Health : Perceptron, LogisticRegression, RandomForest et SVM.

Nous avons réalisé le test en utilisant comme variable cible la colonne text et la colonne humor comme variable explicative. Nous avons d'abord testé les classifieurs sans appliquer de prétraitement aux textes, nous avons simplement vectorisé le texte avec CountVectorizer, ce qui permet de donner une représentation numérique aux textes.

70% des données sont utilisés pour la phase d'apprentissage et 30% sont utilisés pour la phase test.

Nous avons obtenues les résultats suivant:

	Perceptron	Random Forest	SVM	Logistic Regression
Bonnes classifications	36469	36469		36469
Erreurs	3531	3531		3531
Accuracy	0.91	0.91		0.91
Temps d'exécution	0.25 sec	540 sec		1.29 sec

V. Méthodologie

- Student Health

Dans cette partie nous tenterons d'améliorer les résultats obtenus avec SVM pour essayer d'atteindre des résultats similaires à l'article scientifique "Performance Analysis of Student Healthcare Dataset using Classification Algorithm". Nous ferons cela à partir du constat que la classification SVM a donné un poids important à ces features: Health_Risk_Level_Low, Health_Risk_Level_High et Health_Risk_Level_Moderate.
Nous procéderons à une nouvelle classification avec ces deux caractéristiques en tant que variables explicatives, et Stress_Level_Self_Report comme variable cible. Cette démarche vise à visualiser si ces deux caractéristiques choisies comme importantes par SVM, suffisent à elles seules à améliorer la performance de classification, sans inclure les autres variables explicatives.
Le résultat obtenus avec cette méthodologie subissent une légère baisse de performance comparé au résultat obtenus avec l'ensemble des features , avec une précision de 0.58 et un total d'erreurs de 127 sur 300. Cette méthode n'a pas permis d'améliorer la classification.

Bien que les résultats obtenus précédemment ne soient pas satisfaisants, cela reste meilleurs que la performance obtenue pour la classification avec des variables explicatives qui ne sont pas celles choisies par le classifieur. En effet, la classification avec les variables explicatives Heart_Rate et Stress_Level_Self_Report obtient une précision de 44% avec un total d'erreurs de 168 sur 300.
Ces caractéristiques ne sont pas des prédicteurs fiables pour notre variable cible. Contrairement aux caractéristiques utilisées précédemment, cela peut être constaté en analysant l'importance des caractéristiques pour ces deux méthodes.

Classification avec les différents niveaux de santé en variables explicatives :

Feature	**Importance**
Health_Risk_Level_Low	1.333008

Health_Risk_Level_High	0.666992
Health_Risk_Level_Moderate	0.666016

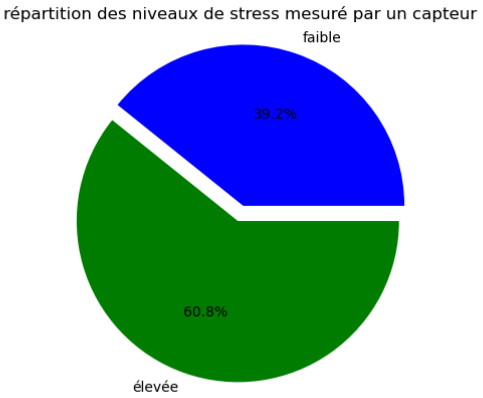
Classification avec deux autres caractéristiques choisis arbitrairement :

Feature	**Importance**
Stress_Level_Self_Report	0.000504
Heart_Rate	0.000478

Nous pouvons constater que les caractéristiques représentant le niveau de santé des étudiants sont plus informatives pour réaliser la classification que les variables Stress_Level_Self_Report et Heart_Rate.

Après cela, nous avons déduit que les résultats de la classification pourraient ne pas être satisfaisant dû aux intervalles de niveaux de stress que nous avons choisis arbitrairement précédemment. En effet le niveau élevé est sous-représenté comparé aux autres niveaux de stress, cette insuffisance notable de données pourrait se traduire par des résultats aussi faibles. Ainsi nous avons changer les intervalles de notre variable cible "Stress_Level_Biosensor":

- Les valeurs de [0-4.5[appartiennent à la catégorie "faible".
- Les valeurs de [4.5-10[appartiennent à la catégorie "élevée".
- Les valeurs de [10-inf] appartiennent à la catégorie "inconnu" car aucun étudiant atteint ce niveau de stress dans notre dataset.



Les résultats obtenus avec cette méthode nous permet d'atteindre une précision de 75% avec un total d'erreur de 74 sur 300.

- Shorts text humor detection

Le classifieur SVM mettait beaucoup trop de temps à afficher un résultat, nous l'avons laissé tourner durant plusieurs heures mais nous n'avons rien obtenu. Pour pouvoir corriger ce problème, nous avons décidé d'appliquer un prétraitement avec Countvectorizer. Les paramètres appliqués sont les suivant :

- Enlever les mots outils (stop words) en anglais : **stop_words = 'english'**
- Prendre seulement les 1000 mots les plus fréquents du corpus : **max_features = 1000**
- Travailler sur des n-grammes de mots (monogrammes et bigrammes) : **ngram_range = (1, 2)**
- Analyser les mots : **analyzer = 'word'**

Après l'application du prétraitement, nous avons pu observer des temps d'exécutions beaucoup moins longs pour les différents classifieurs et nous avons pu obtenir un résultat pour le classifieur SVM.

Notre objectif actuel est d'améliorer le temps d'exécution de SVM donc de le réduire. Pour pouvoir atteindre cet objectif, on a utilisé TfidfVectorizer. Le TF-IDF permet d'accorder un poids aux mots en fonction de leurs importance dans le texte.

Suite à l'application de ce nouveau vectoriseur, nous avons observé des temps d'exécutions beaucoup moins longs pour les différents classifieurs et un résultat encore plus rapide pour le classifieur SVM.

VI. Résultats

Le classifieur SVM a été le plus efficace pour la réalisation de notre travail. SVM signifie "Support Vector Machine" ou "Machine à vecteurs de support" et sont des algorithmes d'apprentissage supervisé utilisés principalement pour des tâches de classification. Leur objectif est de trouver la meilleure frontière possible, appelée hyperplan, qui sépare distinctement les différentes classes de données.

Pour comprendre comment le SVM fonctionne, imaginons un groupe de points répartis en deux catégories distinctes. Le SVM (Support Vector Machine) cherche à trouver l'hyperplan qui sépare ces deux catégories de la manière la plus efficace possible, en maximisant l'espace entre les points les plus proches de chaque côté de l'hyperplan. Ces points, appelés "vecteurs de support", jouent un rôle clé dans cette séparation. En augmentant cette marge, le SVM s'efforce de rendre le modèle plus performant pour prédire correctement de nouvelles données.

- Student Health

```
Resultat de la classification avec tout les caractéristiques en variables explicatives et stress level biosensor en variable cible:
              precision    recall  f1-score   support

      faible       0.00      0.00      0.00       115
       moyen       0.44      1.00      0.61       132
      élevée       0.00      0.00      0.00        53

    accuracy                           0.44       300
   macro avg       0.15      0.33      0.20       300
weighted avg       0.19      0.44      0.27       300

kappa: 0.0
```

Nous pouvons constater que les résultats ne sont pas satisfaisants pour les catégories faible et élevée, seulement la catégorie “moyen” présente des résultats.

```
Resultat de la classification avec Health Risk Level en variable explicative et stress level biosensor en variable cible:
              precision    recall  f1-score   support

      faible       0.90      0.40      0.55       115
       moyen       0.51      0.96      0.67       132
      élevée       0.00      0.00      0.00        53

    accuracy                           0.58       300
   macro avg       0.47      0.45      0.41       300
weighted avg       0.57      0.58      0.51       300

kappa: 0.2568318801568261
```

Ici, les résultats pour les catégories faible et moyen sont plus satisfaisants. De ce fait, la caractéristique “Health_Risk_Level” est un prédicteur fiable pour classifier les niveaux de stress, mais l’accord kappa est faible. En effet, nous pouvons affirmer que la variable “Health_Risk_Level” est la plus fiable car les résultats diminuent lorsque nous tentons de classifier avec d’autres caractéristiques :

```
Resultat de la classification avec Heart Rate et Stress Level Report en variable explicatives et Stress Level Biosensor en variable cible :
              precision    recall  f1-score   support

      faible       0.00      0.00      0.00       115
       moyen       0.44      1.00      0.61       132
      élevée       0.00      0.00      0.00        53

    accuracy                           0.44       300
   macro avg       0.15      0.33      0.20       300
weighted avg       0.19      0.44      0.27       300

kappa: 0.0
```

```
Resultat de la classification après modification des niveaux de stress(variable explicative:Health Risk Level, variable cible: Stress Level Biosensor) :
              precision    recall  f1-score   support

    faible_       0.90      0.40      0.55       115
    élevée_       0.72      0.97      0.83       185

    accuracy                           0.75       300
   macro avg       0.81      0.69      0.69       300
weighted avg       0.79      0.75      0.72       300

kappa: 0.41686367218282117
```

Les résultats sont plus corrects lorsque nous changeons les intervalles des niveaux de stress de manière à fusionner le niveau moyen avec le niveau élevé. Ainsi le nombre minime de données dans la catégorie élevée explique l’obtention de résultats peu satisfaisants lors

des classifications réalisées précédemment. Cependant, l'accord kappa reste tout de même moyen.

Nous n'avons pas obtenu des résultats aussi élevés que dans l'article *"Machine Learning-Based Prediction of Mental Well-Being Using Health Behavior Data from University Students"*, mais nous avons tout de même pu améliorer les résultats que nous avons obtenus en premier lieu.

- Shorts text humor detection

Les résultats suivants sont ceux obtenus après l'application du prétraitement :

	Perceptron	Random Forest	SVM	Logistic Regression
Bonnes classifications	55429	55429	55593	55429
Erreurs	5571	5571	4407	5571
Accuracy	0.91	0.91	0.93	0.91
Temps d'exécution	0.27 sec	467 sec	3682 sec	1.17 sec

On peut observer un temps d'exécution de 3682 secondes soit 1h et 2 minutes environ pour le classifieur SVM. On remarque donc que le prétraitement permet déjà d'obtenir un résultat plus "rapide" qu'initialement.

Par la suite, nous avons voulu améliorer ce temps d'exécution en utilisant un nouveau Vectorizer qui est **TfidfVectorizer**.

On peut constater ci-dessous les résultats que nous avons obtenus avec ce nouveau vectoriseur :

	Perceptron	Random Forest	SVM	Logistic Regression
Bonnes classifications	45512	49396	49396	49396
Erreurs	14488	10604	10604	10604
Accuracy	0.76	0.82	0.82	0.82
Temps d'exécution	0.08 sec	64 sec	336 sec	0.15 sec

VII. Conclusion

Finalement, ce projet nous a permis d'appréhender l'importance du choix de caractéristiques et du prétraitement de données dans la classification. Une bonne compréhension des données peut grandement influencer les résultats des algorithmes. Nous avons exploré divers classifieurs et le SVM s'est démarqué en offrant des résultats plus satisfaisants bien que le temps d'exécution soit plus long. Nous avons pu remédier à cela en manipulant les paramètres de prétraitement pour le dataset Short Text Humor Detection.

L'ajustement des intervalles de la variable cible Stress_Level_Biosensor lors de la classification du dataset Student Health nous a permis d'améliorer nos résultats initiaux , cependant nos résultats n'ont pas atteint ceux des études existantes. Pour remédier à cela, nous aurions pu enrichir le dataset . L'ajout de nouvelles données pourrait permettre aux classifieurs d'apprendre de manière plus approfondie et d'obtenir des résultats plus précis.

Bibliographie

Ziya. “Student-Health-Data.” *Kaggle*, 15 Nov. 2024, www.kaggle.com/datasets/ziya07/student-health-data.

Contractor, Deep. “200k Short Texts for Humor Detection.” *Kaggle*, 2 Apr. 2022, www.kaggle.com/datasets/deepcontractor/200k-short-texts-for-humor-detection?resource=download.

“Onehotencoder.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

“NumPy: Numpy.Hstack() Function.” *W3resource*, www.w3resource.com/numpy/manipulation/hstack.php.

“Pandas.Cut.” *Pandas.Cut - Pandas 2.2.3 Documentation*, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.cut.html.

“Countvectorizer.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. Accessed 17 Dec. 2024.

“TfidfVectorizer.” *Scikit*, scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed 17 Dec. 2024.

Abdul Rahman, Hanif, et al. “Machine Learning-Based Prediction of Mental Well-Being Using Health Behavior Data from University Students.” *MDPI*, Multidisciplinary Digital Publishing Institute, 10 May 2023, www.mdpi.com/2306-5354/10/5/575.

“Support Vector Machine (SVM) Algorithm.” *GeeksforGeeks*, 10 Oct. 2024, www.geeksforgeeks.org/support-vector-machine-algorithm/?ref=gcse_outind#how-does-support-vector-machine-algorithm-work.

“SVM Classifier et Rbf Kernel - Comment Créer de Meilleurs Modèles En Python.” *ICHI.PRO*, 17 Jan. 2021, ichi.pro/fr/svm-classifier-et-rbf-kernel-comment-creer-de-meilleurs-modeles-en-python-127248222170971.