Classification de patients souffrant d’un cancer.

Avec environ 54 062 nouvelles personnes touchées chaque année, le cancer du sein est le plus répandu des cancers féminins mais également masculin. Près d'une femme sur neuf sera concernée au cours de sa vie, le risque augmentant avec l'âge. Moins de 10% des cancers du sein surviennent avant 40 ans. L’incidence augmente ensuite régulièrement jusqu’à 65 ans.
Après avoir doublé entre 1980 et 2005, l'incidence semble désormais en phase de stabilisation. Plus encourageant encore, la mortalité (nombre de décès/an) n'a, elle, pas augmenté depuis les années 80. Le résultat d'énormes progrès, tant au niveau du dépistage que de la prise en charge médicale de la maladie. Pour preuve, aujourd'hui, plus de 3 cancers du sein sur 4 sont guéris en sachant que tous les types de cancers n’ont pas le même pronostic ! Ces scores encourageants sont le fruit de l’effort de la médecine préventive mais également de la capacité du corps médical à prendre en charge rapidement les patients. Ce sujet vous touche particulièrement et vous souhaitez aider le corps médical dans son processus de dépistage.

Ce matin vous avez décidé de développer un classifieur vous permettant d’identifier rapidement la gravité du cancer et ainsi de distinguer si les cellules cancéreuses sont bénignes ou malignes.

Le dataset initial a été créé dans le but de prédire si les cellules cancéreuses sont bénignes ou malignes.

Contenu du fichier de données

Informations sur les attributs:
Sample code number: id number
Clump Thickness: 1 - 10
Uniformity of Cell Size: 1 - 10
Uniformity of Cell Shape: 1 - 10
Marginal Adhesion: 1 - 10

Single Epithelial Cell Size: 1 - 10
Bare Nuclei: 1 - 10
Bland Chromatin: 1 - 10
Normal Nucleoli: 1 - 10
Mitoses: 1 - 10

Predicted class:
2 for benign and 4 for malignant

Cet ensemble de données provient de Original Wisconsin Breast Cancer Database

Pour réaliser cette classification de patients porteurs de cellules cancéreuses bénignes ou malignes, nous allons utiliser un algorithme kNN.
Le classificateur kNN ou k-Nearest Neighbours est un algorithme d'apprentissage automatique très simple et facile à comprendre. Le but de ce brief est de mettre en place un classificateur k Nearest Neighbours pour classer les patients souffrant de cancer du sein.

1. Introduction à l'algorithme k Nearest Neighbours

Dans le domaine de l'apprentissage automatique, k Nearest Neighbours ou kNN est le plus simple de tous les algorithmes d'apprentissage automatique. Il s'agit d'un algorithme non paramétrique utilisé pour les tâches de classification et de régression. “Non paramétrique” signifie qu'aucune hypothèse n'est requise pour la distribution des données. Ainsi, le kNN ne nécessite aucune hypothèse sous-jacente. Dans les tâches de classification et de régression, l'entrée se compose des k exemples d'apprentissage les plus proches dans l'espace des fonctionnalités. La sortie dépend du fait que le kNN est utilisé à des fins de classification ou de régression.

Dans la classification kNN, la sortie est une appartenance à une classe. Le point de données donné est classé en fonction de la majorité du type de ses voisins. Le point de données est affecté à la classe la plus fréquente parmi ses K voisins les plus proches. Habituellement, k est un (petit) entier positif. Si k = 1, ainsi le point de données est simplement affecté à la classe de ce seul voisin le plus proche.

Dans la régression kNN, la sortie est une valeur de propriété pour l'objet. Cette valeur est la moyenne des valeurs des K voisins les plus proches.

L'algorithme kNN est un type d'apprentissage basé sur des instances ou d'apprentissage paresseux. Apprentissage paresseux signifie qu' aucun point de données d'entraînement pour la génération du modèle est nécessaire. Toutes les données d'entraînement seront utilisées dans la phase de test. Cela rend l’apprentissage plus rapide et les tests plus lents et plus coûteux. Ainsi, la phase de test nécessite plus de temps et de ressources mémoire.

Avec un kNN, les voisins sont extraits d'un ensemble d'objets pour lesquels la classe ou la valeur de la propriété de l'objet est connue. Cela peut être considéré comme l'ensemble d'apprentissage pour l'algorithme kNN, bien qu'aucune étape d'apprentissage explicite ne soit requise. Pour un algorithme kNN de classification et de régression, nous pouvons

attribuer un poids aux contributions des voisins. Ainsi, les voisins les plus proches contribuent davantage à la moyenne que les plus éloignés.

2. k Intuition des voisins les plus proches

L'intuition de l'algorithme kNN est très simple à comprendre. Il calcule simplement la distance entre un point de données d'échantillon et tous les autres points de données d'entraînement. La distance peut être la distance euclidienne ou la distance de Manhattan. Ensuite, il sélectionne les k points de données les plus proches où k peut être n'importe quel entier. Enfin, il affecte le point de données d'échantillon à la classe à laquelle appartiennent la majorité des k points de données.

Pour comprendre son fonctionnement, supposons que nous ayons un ensemble de données avec deux variables classées comme rouge et bleu.

Dans l'algorithme kNN, k est le nombre de voisins les plus proches. Généralement, k est un nombre impair car il aide à décider de la majorité de la classe. Lorsque k = 1, alors l'algorithme est connu comme l'algorithme du plus proche voisin.

Maintenant, nous voulons classer un nouveau point de données X en classe Blue ou en classe Red. Supposons que la valeur de k soit 3. L'algorithme kNN commence par calculer la distance entre X et tous les autres points de données. Il trouve ensuite les 3 points les plus proches avec le moins de distance au point X.

Dans la dernière étape de l'algorithme kNN, nous affectons le nouveau point de données X à la majorité de la classe des 3 points les plus proches. Si 2 des 3 points les plus proches appartiennent à la classe Rouge tandis que 1 appartient à la classe Bleu, alors nous classons le nouveau point de données comme Rouge.

3. Comment décider du nombre de voisins en kNN

Lors de la construction du modèle de classificateur kNN, une question qui me vient à l'esprit est de savoir quelle devrait être la valeur des voisins les plus proches (k) qui donne la plus grande précision. C'est une question très importante car la précision de la classification dépend de notre choix de k.

Le nombre de voisins (k) en kNN est un paramètre que nous devons sélectionner au moment de la construction du modèle. La sélection de la valeur optimale de k en kNN est le problème le plus critique. Une petite valeur de k signifie que le bruit aura une influence plus élevée sur le résultat. Ainsi, la probabilité de surajustement est très élevée. Une valeur élevée de k rend la construction du modèle kNN coûteuse en termes de temps. De plus, une valeur élevée de k aura une limite de décision plus lisse, ce qui signifie une variance plus faible mais un biais plus élevé.

Les data scientists choisissent une valeur impaire de k si le nombre de classes est pair. Nous pouvons appliquer la méthode du coude pour sélectionner la valeur de k. Pour optimiser les résultats, nous pouvons utiliser la technique de validation croisée. En utilisant la technique de validation croisée, nous pouvons tester l'algorithme kNN avec différentes

valeurs de k. Le modèle qui donne une bonne précision peut être considéré comme un choix optimal. Cela dépend des cas individuels et le meilleur processus consiste parfois à parcourir chaque valeur possible de k et à tester notre résultat.

Si vous n'avez encore pas tout compris, pas de panique, ce tuto va vous présenter l'algorithme KNN

https://mrmint.fr/introduction-k-nearest-neighbors

Ressources

1. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
2. https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
3. http://dataaspirant.com/2016/12/23/k-nearest-neighbor-classifier-intro/
4. https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
5. https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/4022441-entrainez-votre-premier-k-nn
6. https://fr.wikipedia.org/wiki/M%C3%A9thode_des_k_plus_proches_voisins#:~:text=En%20reconnaissance%20de%20forme%2C%20l,la%20classification%20et%20la%20r%C3%A9gression.&text=en%20r%C3%A9gression%20k%2DNN%2C%20le,des%20k%20plus%20proches%20voisins.
7. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Pour aller plus loin

8. Hands on Machine Learning with Scikit-Learn and Tensorflow by Aurélién Géron
9. Introduction to Machine Learning with Python by Andreas C. Müller and Sarah Guido
10. Udemy course – Machine Learning – A Z by Kirill Eremenko and Hadelin de Ponteves

Pour bien comprendre le fonctionnement d'un KNN, nous allons suivre les étapes suivantes :

1. **Réaliser l'analyse exploratoire des données**
2. **Visualiser les données**
3. **Déclarer le vecteur de caractéristiques et la variable cible**
4. **Divisez les données en un ensemble d'entraînement et de test séparé**
5. **Feature engineering (Ingénierie des fonctionnalités) :** processus de transformation des données brutes en fonctionnalités utiles qui nous aident à mieux comprendre notre modèle et à augmenter sa puissance prédictive : tips : utiliser .isnull().sum()
   Nous supposons que des données sont manquantes aléatoirement. Deux méthodes peuvent être utilisées pour imputer les valeurs manquantes. L'une est l'imputation moyenne ou médiane et l'autre est l'imputation par échantillon aléatoire. Lorsqu'il y a des valeurs aberrantes dans l'ensemble de données, nous devrions utiliser

l'imputation médiane. Vous pouvez donc utiliser des mécanismes d'imputation médiane parce que l'imputation médiane est robuste aux valeurs aberrantes. L'imputation doit être effectuée sur l'ensemble d'apprentissage, puis propagée vers l'ensemble de test. Cela signifie que les mesures statistiques à utiliser pour remplir les valeurs manquantes à la fois dans le train et dans l'ensemble d'essai doivent être extraites de l'ensemble de train uniquement. Ceci pour éviter le surajustement.

6. **Feature scaling**
7. **Ajuster le classificateur à l'ensemble d'entraînement**
8. **Prédire les résultats du test** (predict() et predict_proba())
9. **Vérifiez le score de précision** : Comparez la précision du modèle avec une précision nulle. La précision du modèle sur ce jeu de données devrait être très élevée. Mais, nous ne pouvons pas dire que notre modèle est très bon basé seulement sur cette prédiction. Nous devons le comparer avec la précision nulle. La précision nulle est la précision qui pourrait être obtenue en prédisant toujours la classe la plus fréquente. Vous devez donc vérifier la distribution des classes dans l'ensemble de test. On vérifie donc dans un premier temps la distribution des classes dans l'ensemble de test. (Chez moi les occurrences de la classe la plus fréquente sont 85. je calcule une précision nulle en divisant 85 par le nombre total d'occurrences. le score de précision nul est de 0,6071)
10. **Reconstruire le modèle de classification kNN en utilisant différentes valeurs de k** (vous pouvez par exemple augmenter la valeur de k et voir son effet sur la précision)
11. **Matrice de confusion.** La matrice de confusion vous donnera une image claire de la performance du modèle de classification et des types d'erreurs produites par le modèle. Elle affiche un résumé des prévisions correctes et incorrectes ventilées par catégorie.
12. **Métriques de classification.** rapport de classification : affiche les scores de précision, de rappel, de f1 et de support du modèle

    Nous pouvons classer les observations par probabilité de cancer bénin ou malin.

    Prédisez les probabilités. Pour cela, choisissez la classe avec la probabilité la plus élevée

    Niveau de seuil de classification : Il existe un seuil de classification de 0,5.

    Classe 4 - la probabilité de cancer malin est prédite si la probabilité est> 0,5.

    Classe 2 - la probabilité de cancer bénin est prévue si la probabilité <0,5.
13. **ROC - AUC.** ROC Curve est un autre outil permettant de mesurer visuellement les performances du modèle de classification. ROC Curve est l'acronyme de Receiver Operating Characteristic Curve. Une courbe ROC est un graphique qui montre les performances d'un modèle de classification à différents niveaux de seuil de classification.La courbe ROC trace le taux de vrais positifs (TPR) par rapport au taux de faux positifs (FPR) à différents niveaux de seuil. Le taux vrai positif (TPR) est également appelé rappel. Il est défini comme le rapport de TP à (TP + FN). Le taux de faux positifs (FPR) est défini comme le rapport entre FP et (FP + TN).

    ROC AUC signifie Receiver Operating Characteristic - Area Under Curve. C'est une technique pour comparer les performances du classificateur. Dans cette technique, nous mesurons l'aire sous la courbe (AUC). Un classificateur parfait aura une AUC ROC égale à 1, tandis qu'un classificateur purement aléatoire aura une AUC ROC égale à 0,5.

    Ainsi, ROC AUC est le pourcentage du tracé ROC qui se trouve sous la courbe.

14. **Utiliser la validation croisée K-Fold** : Il s'agit d'une technique de validation croisée pour améliorer les performances du modèle. La validation croisée est une méthode statistique d'évaluation des performances de généralisation. Elle est plus stable et plus approfondie que l'utilisation d'un fractionnement train-test pour évaluer les performances du modèle.
15. **Résultats et conclusion**
16. **Bonus : Implémenter votre propre algorithme KNN**. Vous pouvez vous appuyer sur cet algorithme :

    Proposition non optimisée de l'algorithme kNN
    1. Charger les données
    2. Initialiser k au nombre de plus proches voisins choisi
    3. 3. Pour chaque exemple dans les données:
       3.1 Calculer la distance entre notre cible et l'observation itérative actuelle de la boucle depuis les données. : -> A reformuler …
       3.2 Ajouter la distance et l'indice de l'observation concernée à une collection ordonnée de données
    4. Trier cette collection ordonnée contenant distances et indices de la plus petite distance à la plus grande (dans ordre croissant).
    5. Sélectionner les k premières entrées de la collection de données triées (équivalent aux k plus proches voisins
    6. Obtenir les étiquettes des k entrées sélectionnées
    7. Si régression, retourner la moyenne des k étiquettes
       Si classification, retourner le mode (valeur la plus fréquente/commune) des k étiquettes
17. **Bonus 2 : Comparer les résultats avec un autre algorithme de classification**