Brief : Entraînez votre premier k-NN Application Test De Personnalités

Fevrier 2022

Sommaire :

Contexte du projet :

Cet atelier a pour objectif de vous initier aux techniques de classification supervisé sous Python, plus précisément, nous allons manipuler le classifieur KNN, autrement dit : K plus proche voisin.

Dans cet atelier, nous appliquons la classification pour prédire comment une personne gère son stress.

Challenges :

- Collection d'une base de données.
- Analyse, prétraitement et visualisation des données.
- Préparation des données pour l'apprentissage.
- Conception d'un modèle KNN.
- Choix de la meilleure configuration pour le modèle KNN.
- Vérification de l'efficacité de ce modèle à des nouvelles données.
- Intégration de ce modèle dans l'application de test de personnalité.

Phases du projet (3 parties)

Partie 1 : Base de données, Analyse, Prétraitement et Préparation

Un dossier nommé « Run_Questionnaire » vous sera transmis, vous êtes censés à comprendre et maitriser le contenu et le fonctionnement de ce dossier (Votre fiche de lecture doit être notée dans le compte rendu final).

Exécutez le Notebook « Run.ipynb » pour collecter votre DataSet.

Instruction

- Chacun exécutera 10 fois le Notebook (ne pas trop concentrer sur les questions, et changer vos réponses à chaque fois). Faites attention, il y a des réponses par a, b ou c et d'autre par 1, 2 ou 3.
- Pour traiter tous les cas possibles, vous êtes obligés de mettre des réponses erronées (hors proposition, exemple de réponse : w ou h ou 9 ou pas de réponse carrément).
- Codez un script python qui permet de regrouper les DataSets de chacun en une seule DataSet.

Appliquez les traitements nécessaires pour préparer la DataSet en utilisant Numpy et Pandas, (vous pouvez trouver un référentiel sur ressource) (présenter votre pipeline dans le compte rendu). NB. Le résultat de classification dépond essentiellement de la qualité du prétraitement.

Partie 2 : Développement et entraînement d'un modèle KNN

La technique de classification KNN est considérée comme la technique la plus simple pour appliquer la classification supervisée, tout simplement, une nouvelle donnée de test sera classée comme la majorité de ses voisins (la distance la plus proche). À la suite de votre recherche sur le principe de KNN, nous développons notre modèle KNN. Pour cela :

KNN From Scratch

- Préparez une fonction permettant de calculer les 3 différentes distances : Euclidean, Manhattan et Minkowski, (def distance(metric=' Euclidean', **kargs)).
- Codez l'algorithme de KNN sous forme une fonction (def KNN(Data_Test, Data_Train, Label_Train, k=1, **kargs)) qui :
- Calcul la distance entre Data de test et Data d'apprentissage.
- Trouve la/les distances plus proche de « k » voisins.
- Classe Data de test selon la classe majoritaire de « k » voisins.
- Retourne la classe de Data Test.
- Réalisez des expérimentations en variant la distance et le nombre de « k ».
- Calculez les performances (exemple : Acc) et tracez la courbe de performance de chaque expérimentation. (Les résultats avec interprétation/argumentation doivent figurer dans le notebook comme dans le compte rendu).

KNN Sklearn

La bibliothèque Sklearn propose un panel des techniques de classification, y compris le KNN.

- Dans cette étape, vous êtes orientés vers la classe « sklearn.neighbors » pour maitriser les paramètres et les options possibles.
- Vous êtes censés à préparer un modèle performant pour notre application tout en respectant les consignes de la conception d'un modèle IA (Data préparée, K-fold

validation, hyperparamètre, Gridsearch). (N'oubliez pas de présenter une comparaison entre KNN From Scratch et KNN Sklearn dans le compte rendu).

Partie 3 : Mettre en place la solution dans l'application de test de personnalité

Utilisez la fonction « joblib » pour enregistrer votre modèle, une fois vous avez préparé votre meilleur modèle de classification Faites intégrer cette solution à l'Application

Test de Personnalité et adapter l'application pour comparer le résultat avec et sans IA.

Liste des documents présents sur le repository github

https://github.com/adthw/modele_KNN.git

- Un Notebook bien structuré/organisé qui réalise les différentes étapes de ce projet.
- Un Notebook de l'application adaptée qui affiche les résultats avec et sans IA.
- Un Readme.md pour mettre en avant votre projet (exemple : des fonctions à définir, analyse, réalisation ...).
- Un compte rendu.
- Les Data Sets et les fichiers nécessaires.

Analyse et construction du projet :

KNN from Scratch :

Création des données :

Le fichier Test.py est importé dans l'application run.ipynb. Les réponses sont enregistrées, les données sont traitées, un score est établi en fonction d'un algorithme qui attribue les points. Les réponses et le traitement sont enregistrés dans un fichier csv. Les fichiers csv de plusieurs utilisateurs sont ensuite mis à disposition afin de construire un dataset.

Fusion des csv et construction d'un dataset commun :

Les fichiers csv sont importés dans un dossier avec l'aide du module OS de python. Le module Pandas est utilisé ensuite pour concaténer les fichiers importés dans dataframe enregistré dans nouveau fichier csv.

Choix dans le traitement de données :

Remplacement habituel pour transformer les format object en numériques afin de pouvoir effectuer des calculs

J'ai également décidé de ne pas supprimer les NaN et des les remplacer par la médiane car l'échantillon n'est pas très très peuplé.

Approche algorithmique :

La distance : Manathan et Euclidean étant des dérivées de la distance universelle j'ai utilisé deux manières de calculer la distance. L'une proche de sklearn avec les noms des distances et Minkowski p toujours égal à 3 et l'autre en définissant le paramètre p directement (si p=1 nous sommes dans le cas euclidean et si p=2 nous sommes dans le cas Manathan).

On voit qu'augmenter la valeur de p ne change pas les résultats.

On crée la fonction KNN et on teste avec les calculs de distance différents. On voit que les résultats ne dépassent pas les 24%. Peut-être est-ce dû aux données valides trop peu présentes.

KNN from Sklearn :

Utilisation de la grid Search :

La fonction best param nous retourne : {'metric': 'minkowski', 'n_neighbors': 17}

Création du modèle et application sur les données de Test : 0.7794117647058824

Résultats :

Utilisation de Joblib :

L'utilisation de joblib pour la création du programme final permet de sauvegarder des structures de données dans un fichier pour pouvoir les recharger après.

Conclusion :

Intéressant d'entrer dans la construction du programme de A à Z, comparer les modes de calcul, utiliser GridSearch pour trouver les bons paramètres, construire notre modèle à partir de ceux-ci et arriver jusqu'à l'application finale. Par manque de temps je n'ai pu tester dans le détail le modèle pour la prediction directement à partir du modèle.