The Cancer Genome Atlas Pan-Cancer analysis project

L'objectif de ce projet est d'utiliser les techniques de régression pour prédire une valeur continue en utilisant les données du projet d'analyse pan-cancer du Cancer Genome Atlas. Ce projet vise à étudier les génomes des cancers pour mieux comprendre les caractéristiques moléculaires des cancers et aider à développer de nouvelles stratégies de traitement. Les données de ce projet sont collectées à partir de différents types de tumeurs et sont stockées dans un fichier CSV appelé "data.csv". Le but de ce projet est de former des modèles de régression en utilisant ces données pour prédire une valeur continue représentant une caractéristique moléculaire spécifique des cancers.

Importation des données:

Les données de génome de cancer sont importées à partir d'un fichier CSV nommé "data.csv". Elles sont ensuite stockées dans un DataFrame Pandas appelé "data". Ce DataFrame représente le jeu de données sur lequel les modèles de régression seront formés.

Chargement des étiquettes:

Les étiquettes des données de génome de cancer sont également importées à partir d'un fichier CSV nommé "labels.csv". Elles sont ensuite stockées dans un DataFrame Pandas appelé "labels". Ces étiquettes représentent les variables cibles pour les modèles de régression.

Encodage de catégorie:

La colonne "sample_id" dans les DataFrames "data" et "labels" est encodée à l'aide de la classe LabelEncoder de scikit-learn. Cette étape de prétraitement est nécessaire pour convertir les valeurs de catégories en entiers utilisables par les algorithmes d'apprentissage automatique. Les colonnes encodées sont ensuite imprimées pour vérifier si l'encodage est correct.

Fusion de données et d'étiquettes:

Les DataFrames "data" et "labels" sont fusionnés en un seul DataFrame appelé "merged_data". Cette fusion permet d'avoir à disposition les variables de caractéristiques et les variables cibles dans un seul DataFrame pour la formation des modèles.

Analyse des données:

Une analyse basique et statistique est effectuée sur les DataFrames "data", "labels" et "merged_data". Cependant, cette partie du code est commentée et ne produit pas de résultats utilisables. Il est cependant important de réaliser cette étape pour comprendre les distributions des variables et les éventuelles relations entre elles.

Sélection de fonctionnalités et de cibles:

La colonne "Class" du DataFrame "merged_data" est supprimée et utilisée comme variable cible "y". Les colonnes restantes sont stockées en tant que variables de caractéristiques dans "x". Cette étape est cruciale pour sélectionner les variables les plus pertinentes pour la formation des modèles de régression.

Fractionnement des données:

Les variables de caractéristiques et la variable cible sont ensuite divisées en ensembles d'apprentissage et de test à l'aide de la fonction "train_test_split" de scikit-learn. Cette division permet de mesurer la performance des modèles formés sur des données indépendantes.

Déploiement des modèles:

Ce code présente une implémentation de 4 modèles de régression différents en utilisant des données de génome de cancer. Les modèles sont utilisés pour prédire une variable cible continue à partir d'un ensemble de variables de caractéristiques. Les 4 modèles incluent la régression linéaire simple (SLR), la régression linéaire multiple (MLR), la régression de crête (RR) et la régression de lasso (LR).

La régression linéaire simple est un modèle de régression utilisant une seule variable de caractéristique pour prédire la variable cible. La régression linéaire multiple utilise plusieurs variables de caractéristiques pour prédire la variable cible. La régression de crête est un modèle de régression qui utilise une fonction polynomiale pour modéliser les relations complexes entre les variables de caractéristiques et la variable cible. La régression de lasso est un modèle de régression qui utilise une régularisation L1 pour sélectionner les variables de caractéristiques les plus pertinentes pour la prédiction.

Tous ces modèles seront formés sur les données d'entraînement (x_train, y_train) et seront évalués en fonction de leur performance sur des données de test (x_test, y_test) indépendantes.

Evaluation:

Voici les résultats dans un tableau pour une meilleure visualisation :

	Model score	Cross-validation	Mean squared error
Simple linear regression	0.98	0.98	0.05
Multiple Linear Regression	0.98	0.98	[0.98,0.99,0.99,0.98, 0.99]
Ridge regression	0.98	0.98	[0.98,0.99,0.99,0.98, 0.99]
Lasso Regression	0.92	0.94	[0.94,0.93,0.95,0.93, 0.94]

Pour évaluer les modèles, nous utilisons 2 métriques différentes: l'erreur quadratique moyenne (mean squared error) et la validation croisée (cross-validation). L'erreur quadratique moyenne mesure la distance entre les valeurs prédites et les valeurs réelles pour les données de test (y_test, y_predict). La validation croisée mesure la performance du modèle en utilisant plusieurs séparations aléatoires des données d'entraînement en ensembles d'entraînement et de test.

Le paramètre alpha dans les modèles de régression de crête et de lasso contrôle la complexité du modèle et aide à prévenir le surapprentissage en imposant une régularisation sur les coefficients de régression. Le choix de 0,5 pour alpha dans ce code est arbitraire et peut nécessiter un ajustement pour obtenir les meilleurs résultats.

Les résultats de nos quatre modèles de régression montrent une performance relativement élevée, avec des scores de précision allant de 0.92 à 0.98. Cependant, il est important de noter que la régression de lasso a un score de précision inférieur aux autres modèles, ainsi qu'une erreur quadratique moyenne plus élevée.

En regardant les scores de validation croisée, nous pouvons voir que les modèles de régression linéaire simple et multiple ont des scores presque identiques, tandis que la régression de crête et la régression de lasso ont des scores légèrement inférieurs.

En résumé, nous pouvons dire que la régression linéaire simple et multiple ont des performances comparables, tandis que la régression de crête et la régression de lasso ont des performances légèrement inférieures.