Machine Learning Project 1:

Overfitting, Underfitting and Metaparameters

1) Training and testing datasets

Pour aborder les tâches de ce premier projet, nous commençons par importer les données depuis la librairie scikit-learn. Le dataset est ensuite divisé en deux ensembles : un ensemble d'entraînement, utilisé pour l'apprentissage du modèle, et un ensemble de test, destiné à évaluer ses performances. Dans ce rapport, l'erreur de généralisation du modèle sera estimée à partir des performances obtenues sur cet ensemble de test indépendant. Nous avons choisi d'allouer 20 % des données à l'ensemble de test et 80 % à l'ensemble d'entraînement pour la division du dataset Diabetes.

Par ailleurs, selon la source Scikit-learn, les données ont déjà été normalisées, il n'y a donc pas de normalisation à réaliser pour nos données.

2) Writing our own model

Avec les données maintenant prêtes à être manipulées, nous pouvons commencer à «construire» notre premier modèle. Pour ce modèle de régression linéaire, nous allons le sur base du critère de la Mean Square Error (MSE). Ce critère aura pour objectif d'être minimisé, car il correspond à la moyenne des écarts au carré entre la valeur réelle que nous essayons de prédire et la prédiction du modèle. Lorsque nous utilisons la MSE comme critère d'erreur, en plus des hypothèses faites pour la régression linéaire, cela revient à supposer que les instances sont i.i.d. et que le bruit que nous observons est gaussien. De plus, le carré de la MSE amplifie les grands écarts, ce qui rend ce critère très sensible aux valeurs extrêmes (outliers).

Pour trouver le modèle qui minimise la MSE, nous allons utiliser la technique analytique de la pseudo-inverse, qui permet de déterminer les poids optimaux de la fonction linéaire en une étape. Afin de pouvoir calculer cette matrice de poids, nous allons utiliser la librairie NumPy, qui permettra d'effectuer les calculs.

Pour commencer, nous allons artificiellement ajouter une colonne remplie de 1 dans nos caractéristiques afin de permettre le calcul du biais. Pour rappel, le calcul des coefficients optimaux avec la MSE est donné par la formule $\beta=(X^TX)^{-1}X^Ty$. Avec la méthode de la pseudo inverse, ceci correspond à faire $\beta=X^+y$, X^+ étant la pseudo inverse de X.

En utilisant un random_state égal à 19 pour séparer les données, nous obtenons les coefficients suivants :

	VALEURS
INTERCEPT	151.667479
AGE	-46.289807

SEX	-186.668002
BMI	517.427128
BP	280.236908
S1	-999.938343
S2	645.340859
S3	187.377693
S4	225.611912
S5	870.695971
S6	31.358468

Nous pouvons voir que les coefficients sont relativement grands, ce qui peut être en partie expliqué par la normalisation des données. Nous pouvons également constater que certains facteurs, comme S2 ou le BMI, semblent favoriser la progression de la maladie, tandis que d'autres, avec des coefficients négatifs, comme l'âge ou S1, semblent la diminuer.

Avec notre matrice de coefficients enfin calculée, nous pouvons prédire les valeurs de la progression du diabète sur l'ensemble de test et ensuite calculer la performance de notre modèle en utilisant la métrique R^2. Pour ce modèle, nous obtenons un R^2 de **0,4772**, ce qui signifie qu'environ **47,72 %** de la variance de la variable cible, dans notre cas la progression du diabète, est expliquée par ce modèle. C'est un résultat relativement bon pour un modèle de régression linéaire ce basant uniquement sur la MSE.

3) Lasso models from scikit-learn

Optons à présent pour un modèle de type Lasso de la librairie Scikit-learn. Nous allons d'abord créer et entraîner un modèle avec un alpha de 0,5. L'alpha est un metaparamètre du modèle qui ajuste l'importance donnée à l'erreur par rapport au nombre de coefficients non nuls. Plus l'alpha est grand, plus le modèle favorise la réduction du nombre de coefficients non nuls, c'est-à-dire des modèles plus simples, avec davantage de coefficients égaux à zéro.

Avec ce modèle, nous obtenons un R^2 de 0,4078, ce qui est plus faible que le R^2 obtenu avec notre premier modèle. Cette diminution peut s'expliquer par le fait que le modèle Lasso favorise plusieurs coefficients nuls, ce qui réduit légèrement le « pouvoir » explicatif de notre régression. Ce modèle Lasso est plus simple et moins complexe que le premier, et semble offrir une moins bonne prédiction sur un ensemble de données qu'il n'a jamais vu, ce qui suggère une moins bonne performance et un léger cas d'underfitting.

Voici les coefficients obtenus avec le modèle Lasso :

	VALEURS
INTERCEPT	151.138640

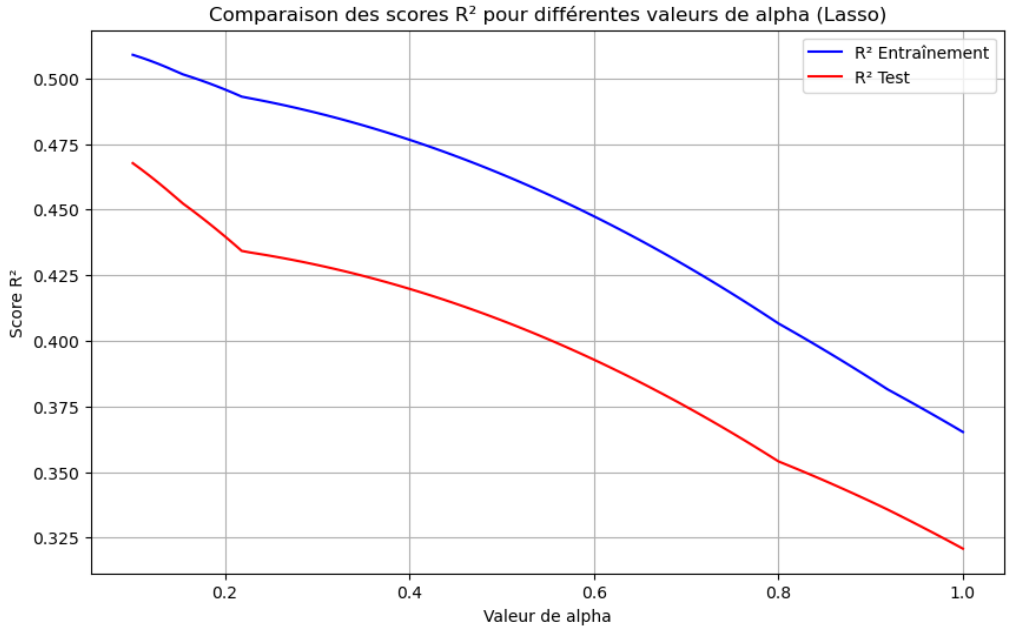
AGE	0.000000
SEX	-0.000000
BMI	456.134098
BP	81.631137
S1	-0.000000
S2	-0.000000
S3	-105.053401
S4	0.000000
S5	439.324290
S6	0.000000

Nous observons que les coefficients associés à l'Âge, au Sexe, à S1, S2, S4 et S6 sont devenus nuls. Quant aux autres coefficients, la constante est presque identique, et les valeurs des autres variables ont légèrement changé, sans grande différence, à l'exception de S3 qui a changé de signe.

4) Comparing models of increasing complexity

Dans cette section nous allons calculer 100 modèles différent avec des alphas allant de 0.1 à 1 et calculer leurs performances de chacun de ceci avec le R^2 sur les données d'entrainement mais aussi sur les données de test.

Voici le graphique comparant les différents R^2 pour différentes valeurs de alpha :



Comme nous pouvons le voir sur le graphique, les valeurs du score de performance R^2 diminuent lorsque la valeur de alpha augmente, tant pour le R^2 d'entraînement que pour le R^2

de test. Pour rappel, plus alpha est grand, plus il y a de coefficients égaux à zéro, ce qui simplifie le modèle. Ce qui est étonnant concernant les résultats obtenus pour les différents modèles, c'est que plus alpha est petit, plus le modèle est complexe. On pourrait s'attendre à un cas d'overfitting, où notre modèle apprendrait trop les données et afficherait de mauvaises performances sur des données qu'il n'aurait jamais vues mais nous pouvons voir que le R^2 issu de l'ensemble de validation augmente aussi. C'est cet indicateur qui nous permet d'estimer notre possible erreur de généralisation et de déterminer si notre modèle apprend trop les données d'entraînement.

Néanmoins, il se peut que nous n'ayons pas eu de chance et que notre séparation des données en un ensemble d'entraînement et un ensemble de test ait été malchanceuse, ce qui pourrait nous conduire à des résultats erronés. C'est pourquoi nous allons procéder à une 10-Fold Cross-Validation pour nous assurer de la robustesse de nos résultats.

5) Choose your model

Lors de la 10-Fold Cross-Validation le meilleure méta-paramètre alpha semble être 0.1. Il est important de noter que nous n'avons tester que 100 valeurs de alpha allant de 0.1 à 1. La performance moyenne des 10 différents modèles est de 0.4592. Lorsque nous entrainons le modèle avec ce méta-paramètre nous obtenons un R^2 de 0.4677 sur l'ensemble de test, ce qui est relativement bon mais reste au même niveau que notre tout premier modèle.

Ce résultat est néanmoins à interpréter avec prudence, car nous ne testons notre modèle que sur un ensemble de test relativement petit. Avec seulement 442 instances que nous divisons en 80-20, il est probable que notre modèle perde en performance s'il est appliqué à une population plus large. Nous sommes donc peut-être en situation d'overfitting.

Malgré cela, notre analyse montre que lorsque nous optons pour des modèles plus simples avec des valeurs d'alpha plus élevées, qui réduisent normalement les cas d'overfitting, nos modèles deviennent de moins en moins performants, tant sur les données d'entraînement que sur les données de test. Cette diminution de performance sur les données de test nous laisse penser que nous n'exploitons pas pleinement le « potentiel » des données, ce qui suggère des cas d'underfitting.

En conclusion, le modèle initial présente de très bonnes performances, et un modèle Lasso avec un alpha de 0,1 offre des résultats similaires. Néanmoins, il faut rester vigilant face à un possible cas de surapprentissage, ce qui souligne l'importance de trouver un équilibre entre complexité et simplicité pour assurer la robustesse du modèle pour la généralisation.