Éléments de statistique
Homework 2 : Estimation par intervalle et régression

1er décembre 2024

Ce devoir porte sur l'estimation par intervalle et la régression. Il vous est demandé de compléter le notebook (`math0487_hw2.ipynb`) fourni séparément. En complément, il est vivement conseillé de créer un fichier Python (`fonctions.py`) contenant diverses fonctions qui vous seront utiles pour ce travail. L'objectif de ce fichier est de vous aider à valider vos implémentations. Des tests automatiques sont disponibles sur Gradescope. Cependant, veuillez noter que ces tests ne couvrent pas nécessairement tous les cas possibles. Passer l'ensemble des tests est un bon indicateur, mais cela ne garantit pas que votre code soit entièrement correct. D'autres tests supplémentaires, notamment pour vérifier le respect des consignes ou détecter le plagiat, pourront être réalisés lors de la correction finale. Notez également que vous disposez d'un nombre illimité de soumissions sur Gradescope.

En ce qui concerne le notebook (`math0487_hw2.ipynb`), vous avez deux options pour utiliser vos fonctions :

— Vous pouvez recoder directement les fonctions à l'intérieur du notebook sans utiliser `fonctions.py` .

— Vous pouvez réutiliser les fonctions définies dans `fonctions.py` en ajoutant `from fonctions import *` dans la cellule dédiée aux imports.

Si vous rencontrez des difficultés à importer `fonctions.py` , vérifiez que ce fichier se trouve bien dans le même répertoire que votre notebook.

Pour développer et tester vos fonctions de votre côté, nous recommandons :

— De les coder d'abord dans le notebook pour faciliter vos tests.

— Ou de les écrire directement dans `fonctions.py` , en ajoutant à la fin un bloc `if __name__ == "__main__":` pour réaliser vos tests. Si vous laissez ces tests dans votre fichier Python, cela ne posera pas de problème lors de votre soumission sur Gradescope.

Les prototypes des fonctions à respecter sont détaillés dans la suite.

Bon travail !

Estimation par intervalle

Les fonctions utilisées dans cette partie doivent renvoyer une liste ou un tuple contenant les limites de votre intervalle.

`get_normal_CI (data, confidence)`
Cette fonction calcule un intervalle de confiance Normal approximé à partir des données se trouvant dans le np.array `data` tout en prenant `confidence` comme valeur de $1 - \alpha$.

`get_boostrap (data, confidence, n_resamples, fun)`
Cette fonction effectue une estimation par intervalle bootstrap en appliquant la méthode du "percentile". L'estimateur doit être calculé à l'aide de la fonction fournie dans `fun`, et `n_resamples` échantillons bootstrap doivent être générés. De plus, le paramètre `confidence` représente le niveau de confiance correspondant à $1 - \alpha$.

Régression linéaire et logistique

`get_linear_model (data, y)`
Cette fonction crée et ajuste un modèle de régression de scikit-learn en l'entraînant sur `data` (`np.array`) et `y` (`np.array`). Elle retourne le modèle ajusté ainsi que les prédictions effectuées sur les données utilisées pour l'entraînement.

`get_residue (y, y_pred)`
Cette fonction calcule et renvoie le résidu entre les vecteurs `y` et `y_pred`, qui sont tous deux des `np.array`. Le vecteur `y` représente les variables observées, tandis que `y_pred` contient les prédictions issues d'un modèle linéaire.

`get_leverage (X)`
Cette fonction retourne le leverage de chaque instance de données (cristallisation) présente dans la matrice `X`, conformément aux équations Normales. `X` est un `np.array` bidimensionnel contenant uniquement les caractéristiques (features), sans inclure la colonne de biais remplie de 1.

`get_specific_residue_leverage (X, y, x_pos, y_pos)`

Cette fonction calcule les résidus et le leverage pour différents points spécifiés dans `x_pos` et `y_pos`. Plus précisément, `x_pos` et `y_pos` contiennent respectivement les positions en abscisse et en ordonnée des points à ajouter individuellement aux données de `X` et `y`. Chaque point de `x_pos` et `y_pos` est intégré séparément au modèle, qui est ensuite réajusté avant de calculer les valeurs souhaitées. Le point en question est ensuite retiré des données. Les résidus et leverage doivent être renvoyés dans le même ordre que les points. Cette fonction est utile pour évaluer le résidu et le leverage de points spécifiques ne faisant pas partie des données initiales.

`get_logistic_model (data, y)`

Cette fonction crée et ajuste un modèle de régression logistique de scikit-learn en l'entraînant sur les données contenues dans `data` (`np.array`) et `y` (`np.array`). Elle retourne le modèle ajusté ainsi que les prédictions générées en appliquant ce modèle sur les données utilisées lors de l'entraînement.

Note : Ces prototypes sont ceux de nos implémentations. Certains arguments ont été introduits afin d'obtenir un code plus ou moins générique. Il se peut que vous n'utilisiez pas certains de ces arguments en fonction de votre implémentation.