Définitions

predictors : Variables d'entrées/*input variables* utilisées dans le cadre d'un problème d'apprentissage supervisé.

Interrogations/Brainstorm

TOUTES LES NOTIONS CI-DESSOUS RESTENT A COMPRENDRE OU VALIDER ! ! !

1. **Inspection / Visualisation des données** :
   - Afin de minimiser le biais du surapprentissage, devrait-on réserver un *test set* avant même de regarder les données ?

     Le simple fait de visualiser les données nous portera à faire des choix. Ces choix intégrés dans notre processus, sont eux-mêmes une forme d'apprentissage.

     D'un autre côté, si on veut un créer un *stratified test set*, représentatif de la population futur, il faut faire une analyse minimum des données. Et même ce *stratified test set*, s'il est conçu à partir d'un sous-ensemble sélectionné de *predictors*, il y a une assomption dans cette sélection. C'est ce que *Géron* fait dans le chapître 2 en créant sont *test set* à l'aide d'une catégorie de revenue.

   - Est-ce plus pertinent dans certains cas d'avoir un test set non stratifié, mais qui couvre mieux l'ensemble des profils d'entrées, par exemple en surreprésentant certains outliers ?

     e.g. Comment mon modèle généralise-t-il pour les cas plus problématique ?

   - Afin de minimiser le biais du surapprentissage, devrait-on réserver un *test set* avant d'effectuer une préselection des *predictors*, ou encore l'étape du *data cleanup*.

     Il y des cas par exemple où si le nombre de *predictors* dépassent largement le nombre d'échantillons du *dataset*, il est impératif de former son *test set* avant de faire une sélection de *predictors*.

     Voir : `https://youtu.be/S06JpVoNaA0`
2. **Étape de nettoyage** :
   - **Données manquantes / valeurs nan** :

     La plupart des modèles mathématiques ne gèrent pas bien les données manquantes. Il faut nettoyer les données pour utiliser ces modèles durant l'étape d'apprentissage. Par contre, il ne faut jamais perdre de vue que ce même modèle utilisé en production pourrait aussi recevoir des entrées incomplètes. Il faut établir une stratégie de nettoyage de données autant pour la étape d'apprentissage que la étape de production.

     Il est donc très important de comprendre en quoi cette stratégie de nettoyage de données affecte le processus d'apprentissage. Comment peut-elle biaiser les résultats ?

     Par exemples :
     - Est-ce que la distribution des échantillons ayant des données manquantes sera la même dans le futur ?
     - Est-ce qu'une donnée manquante est une information pertinente en soi ?

       Exemple : Dans le jeu de données du Titanic, il semble que le fait qu'un passager n'ait pas de numéro de cabine (*nan*), pourrait indiquer qu'il n'ait pas de cabine qui lui ait été attribuée ou réservée. cabine. Peu importe la raison, la donnée manquante a une certaine valeur prédictive en soi.

   - **Stratégies de nettoyage :**
     (a) Élimination de données/*row* : Pour les colonnes où il y a peu de *nan* (*e.g.* 1% à 2% du jeu de données), on peut sans trop de risques éliminer les échantillons/lignes incluant ces *nan*.

     (b) Élimination du *predictor*/*colonne* : Si un *predictor* ne semble pas pertinent, ou qu'il manque trop de données, on peut l'éliminer.

     (c) Stratégie de la moyenne/médiane/mode/"valeur neutre" : On remplace les données manquantes d'une colonne, par la moyenne/médiane/mode/"valeur neutre".

     Si ces données sont en quantités non-négligeagles, cela vient modifier la distribution de la colonne modifiée.

     Dans quelles circonstances corrompt-on l'apprentissage, lorsqu'on floue le modèle lorsqu'on lui indique que pour un échantillon donnée, un de ces *predictors* est égal à la moyenne/médiane/mode, alors qu'en réalité on ne connaît pas la vraie valeur ?

     (d) Ajouter un label *unknown* si le *predictor* est une catégorie : Pour une *predictor* de type catégorie, on peut simplement ajouter un label *unknown*, pour indiquer au modèle qu'on ne connaît la valeur.

     (e) Transformation d'un *predictor* $\in R$ en une catégorie : On transforme notre *predictor* en catégorie et on applique la stratégie précédente.

3. **Feature extraction** :
   - Comment évalue-t-on qu'un *predictor* est pertinent ou pas ?
   - Quels sont les bonnes stratégies pour créer des features ? Ça doit dépendre des domaines...
   - Quand est-ce qu'on a trop de *predictors* ?
   - Quels modèles sont plus/moins sensibles aux nombres de *predictors* qui leurs sont présentés ?
     En d'autres mots est-ce que le modèle est capable de quelconque manière de sélectionner les *predictors* pertinents ?
     Est-ce que le modèle aura tendance à overfitter ou mal généraliser avec trop de *predictors* ?
4. **Model Selection** :
   - Quel est la bonne fonction de perte pertinente à évaluer ?
   - Y-a-t-il plus qu'une métrique á évaluer ? *e.g Recall* vs *Precision*, etc.
   - *Validation Set* vs *Test Set* :
     Dans la plupart des stratégies présentées, on fait une *cross-validation* sur le *training set* pour optimiser les hyperparamétres d'un modèle. L'optimisation des hyperparamétres est une forme d'entrainement au même titre que l'entraînement normal d'un modèle.
     Lors de cette étape, on peut estimer empiriquement la moyenne et la variance de sa fonction de perte, sur la du *fold* réservée à la validation. Mais ces estimations sont généralement biaisées vers le bas, comme toute estimation issue d'un entraînement / fitting.
     Lorsqu'on résout un problème de régression, on peut évaluer la performance finale du modèle via un *Test Set*. On obtiendra un estimation empirique de de la moyenne et la variance de la fonction de perte.
   - Inner/Outer cross-validation :
     Lorsqu'on a la puissance de calcul/le temps pour le faire on peut appliquer une stratégie de *cross-validation* aussi pour le *Test Set*.
     Donc au lieu d'appliquer une stratégie *Hold-Out* en gardant un seul et unique *Test Set* final, on applique aussi un *K-Fold cross-validation* pour le *Test Set*.
     Inner pour la *cross-validation* de chaque modèle.
     Outer pour la *cross-validation* l'évaluation du modèle final.
     Normalement si notre modèle généralise bien, il ressortira gagnant à chaque étape de la *Outer cross-validation*