Machine Learning
Préparation des données et évaluation

Abdelkrime ARIES

Laboratoire de la Communication dans les Systèmes Informatiques (LCSI)

Karima BENATCHBA

Laboratoire des Méthodes de Conception des Systèmes (LMCS)

École nationale Supérieure d'Informatique (ESI, ex. INI), Alger, Algérie

Année universitaire : 2022/2023

Licence : CC-BY 4.0

Attribution 4.0 International (CC BY 4.0)

https://creativecommons.org/licenses/by/4.0/deed.fr

Vous êtes autorisé à :

Partager — copier, distribuer et communiquer le matériel par tous moyens et sous tous formats

Adapter — remixer, transformer et créer à partir du matériel pour toute utilisation, y compris commerciale.

Selon les conditions suivantes :

Attribution — Vous devez créditer l'Œuvre, intégrer un lien vers la licence et indiquer si des modifications ont été effectuées à l'Oeuvre. Vous devez indiquer ces informations par tous les moyens raisonnables, sans toutefois suggérer que l'Offrant vous soutient ou soutient la façon dont vous avez utilisé son Oeuvre.

Pas de restrictions complémentaires — Vous n'êtes pas autorisé à appliquer des conditions légales ou des mesures techniques qui restreindraient légalement autrui à utiliser l'Oeuvre dans les conditions décrites par la licence.

Machine Learning

Préparation des données et évaluation : Introduction

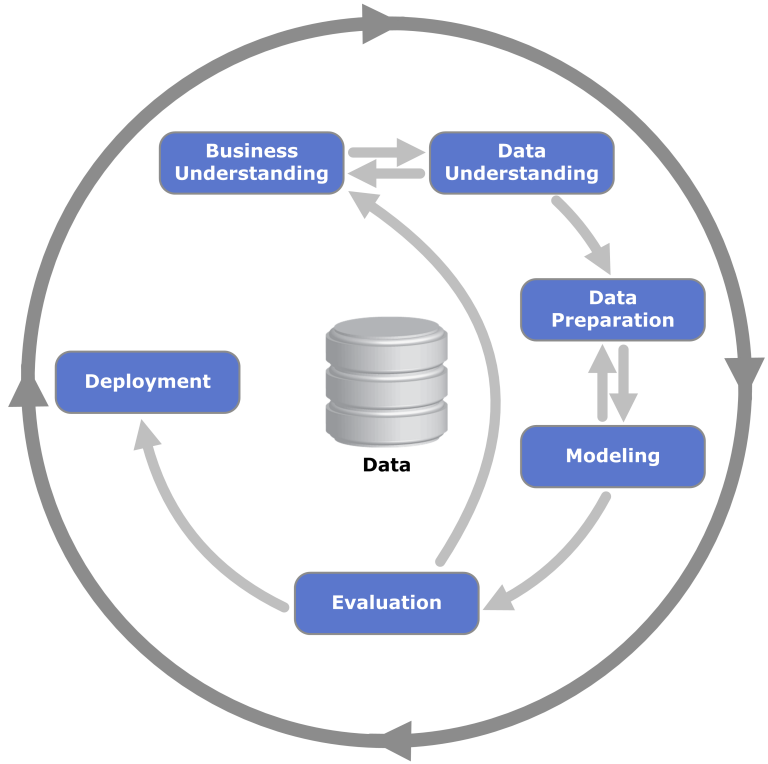


Figure – CRISP-DM (Cross-industry standard process for data mining)

Machine Learning

Préparation des données et évaluation : Plan

Section 1

Collecte de données

Collecte de données

Qualité des données

- Taille : Nombre d'échantillons (enregistrements).
- Nombre et type des caractéristiques (nominales, binaires, ordinales ou continues).
- Erreurs d'annotation.
- Quantité de bruits dans les données : erreurs et exceptions.

Collecte de données

Intégration des données

- Sources des données
  - Données structurées : BD, tabulateurs (CSV, etc.)
  - Données semi-structurées : XML, JSON, etc.
  - Données non structurées : documents textes, images, métadonnées, etc.
- Intégrité des données
  - Conformité des fichiers XML à leurs définitions DTD
  - Séparateurs correctes des fichiers CSV
- Fusionnement des données (joindre les schémas)
  - Problème de nommage : bd1.numclient, bd2.clientid
  - Conflits de valeurs : bd1.taille(cm), bd2.taille(pouces)
  - Redondance : attributs calculés et enregistrements identiques.
  - Types différents des attributs : bd1.temperature (froid, chaud, ...), bd2.temperature (15, 23, ...)

Collecte de données

Annotation (Étiquetage) des données : Annotation interne

- Une équipe chargée par l'annotation
- Meilleure s'il y a suffisamment de ressources humaines, financières et du temps
- Avantages
  - Capacité à suivre le progrès
  - Bonne qualité
- Inconvénients
  - Trop lente : si on gagne de la qualité, on va perdre du temps.

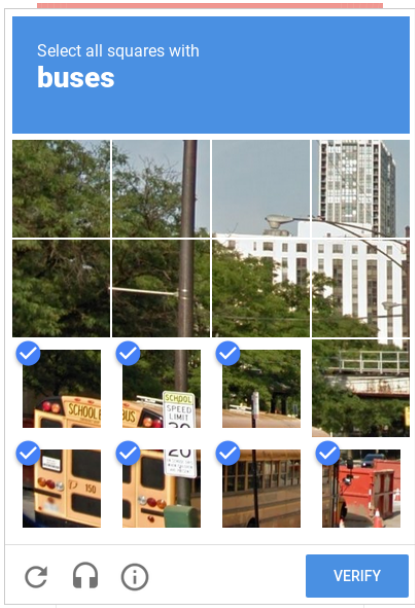
Collecte de données

Annotation (Étiquetage) des données : Externalisation (Outsourcing)

- Embaucher des travailleurs indépendants (freelancers)
- S'il n'y a pas suffisamment de ressources humaines ou du temps
- Étapes
  1. Préparer les données et fixer le temps exigé pour les annoter
  2. Diviser les sur des sous ensembles
  3. Publier des offres d'emploi sur les médias sociaux
- Avantages
  - On connait ceux qu'on a embauché : On peut vérifier leurs compétences
- Inconvénients
  - Préparation des instructions détaillées de l'annotation
  - Temps de soumission et vérification des tâches
  - Création d'un flux de travail : une interface qui aide les annotateurs.

Collecte de données

Annotation (Étiquetage) des données : Crowdsourcing



- Utiliser des plateformes de crowdsourcing.
- Ex. Amazon Mechanical Turk (MTurk) et Clickworker.
- Types de crowdsourcing
  1. Explicite : En demandant directement des contributions
  2. Implicite : En intégrant des tâches dans d'autres formes
     - Tâches inévitables (Ex. reCAPTCHA)
     - Jeux ayant des objectifs (Ex. jeu ESP)
- Avantages
  - Des résultats rapides
  - Coûts abordables
- Inconvénients
  - Qualité des annotations
  - Préparation des instructions détaillées sur le processus d'annotation

Collecte de données

Annotation (Étiquetage) des données : Évaluation (Kappa ; classification)

- Concordance inter-juges (fiabilité inter-évaluateurs)
- Kappa de Cohen
  - Mesure l'agrément entre deux juges et deux catégories
  - $K = \frac{P_o - P_e}{1 - P_e}$
  - $P_o = \frac{a+d}{a+b+c+d}$
  - $P_{oui} = \frac{a+b}{a+b+c+d} * \frac{a+c}{a+b+c+d}$
  - $P_{non} = \frac{c+d}{a+b+c+d} * \frac{b+d}{a+b+c+d}$
  - $P_e = P_{oui} + P_{non}$
  - sklearn.metrics.cohen_kappa_score

		B	
		Oui	Non
A	Oui	a	b
	Non	c	d

- Pi de Scott
  - Comme Cohen mais P_e est calculée différemment (multi-classes)
- Kappa de Fleiss
  - Généralisation de Pi de Scott pour plus de deux évaluateurs

Collecte de données

Annotation (Étiquetage) des données : Évaluation (Corrélation ; régression)

- Pearson
  - Espace continu des valeurs (régression)
  - $r_{Y^{(1)},Y^{(2)}} = \frac{covariance(Y^{(1)},Y^{(2)})}{\sigma_{Y^{(1)}}\sigma_{Y^{(2)}}}$
  - scipy.stats.pearsonr
- Spearman
  - Espace ordinal des valeurs
  - $\rho_{Y^{(1)},Y^{(2)}} = \frac{covariance(rg(Y^{(1)}),rg(Y^{(2)}))}{\sigma_{rg(Y^{(1)})}\sigma_{rg(Y^{(2)})}}$
  - *rg* est la fonction rand ; l'ordre croissant
  - scipy.stats.spearmanr
- Kendall
  - Espace ordinal des valeurs
  - $\tau_{Y^{(1)},Y^{(2)}} = \frac{2}{n(n-1)} \sum_{i<j} signe(y_i^{(1)} - y_j^{(1)}) signe(y_i^{(2)} - y_j^{(2)})$
  - scipy.stats.kendalltau

Collecte de données

Nettoyage des données : Problèmes

- Valeurs omises (données non disponibles)
  - Mauvais fonctionnement de l'équipement
  - Incohérences avec d'autres données et donc supprimées
  - Non saisies car non (ou mal) comprises
  - Considérées peu importantes au moment de la saisie
- Échantillons dupliqués
  - Plusieurs sources de données
- Des mauvaises annotations
  - Incohérence dans les conventions de nommage
- Bruit (erreur ou variance aléatoire d'une variable mesurée)
  - Instrument de mesure défectueux
  - Problème de saisie
  - Problème de transmission

Collecte de données

Nettoyage des données : Solutions possibles

- Valeurs omises (données non disponibles)
  - Suppression ou saisie manuelle
  - Remplacement par une constante globale. Ex., ”inconnu” ou ”0”.
  - Remplacement par la moyenne (numériques), préférence : même classe.
  - Remplacement par la valeur la plus fréquente (nominales).
- Échantillons dupliqués
  - Suppression
- Des mauvaises annotations
  - Mesurer la fiabilité inter-évaluateurs
- Bruit
  - Clustering pour détecter les exceptions
  - Détection automatique des valeurs suspectes et vérification humaine.
  - Lissage des données par des méthodes de régression.

Collecte de données

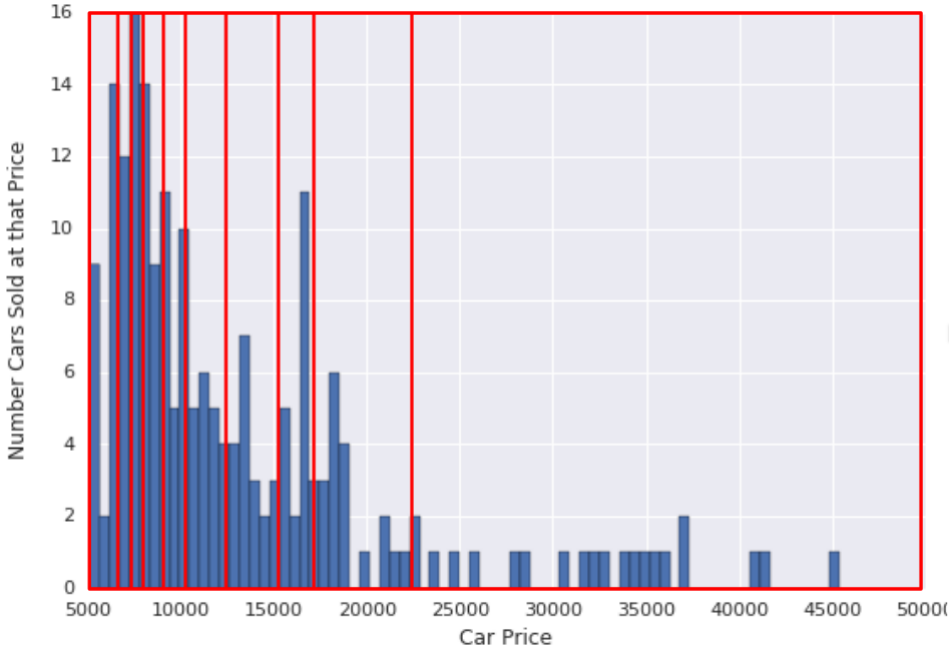
Un peu d'humour

Section 2

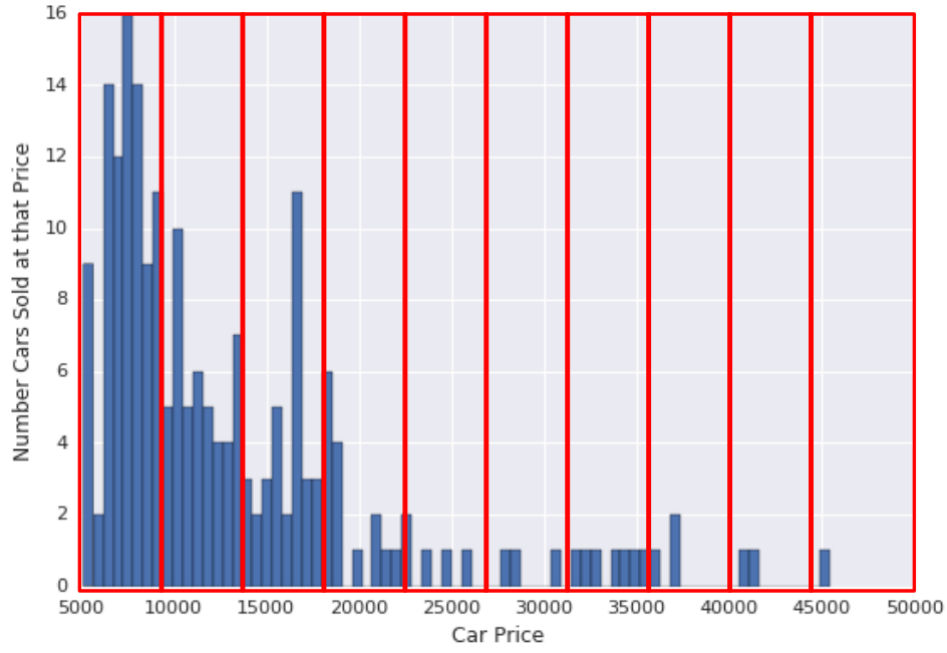
Transformation des données

Transformation des données

Valeurs numériques : Discrétisation par groupement (binning, bucketing)



quantiles



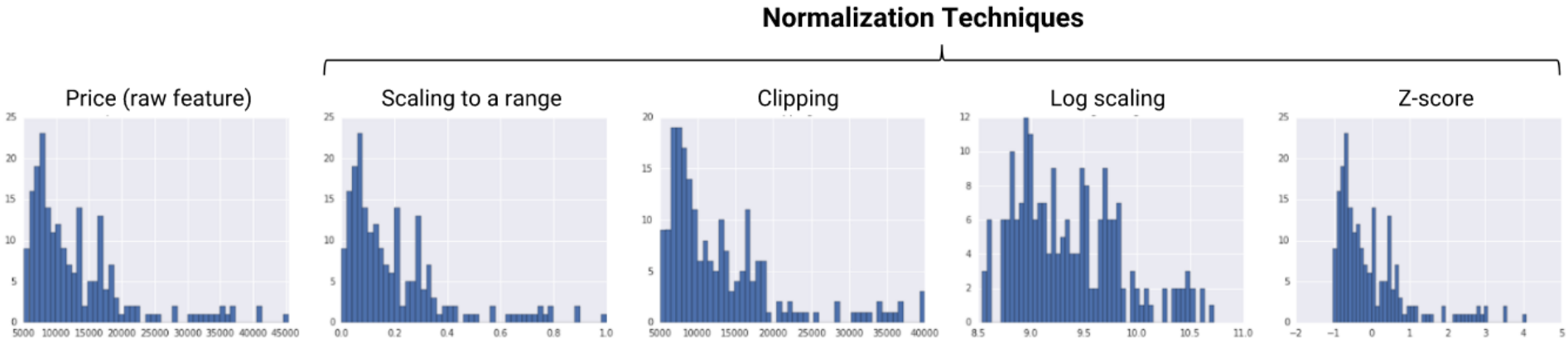
plages identiques

Exemple de discrétisation par regroupement [Google Developers, 2021a]

- lorsqu'il n'y a pas de relation linéaire avec la sortie
- sklearn.preprocessing.KBinsDiscretizer

Transformation des données

Valeurs numériques : Normalisation



Comparaison entre les fonction de normalisation

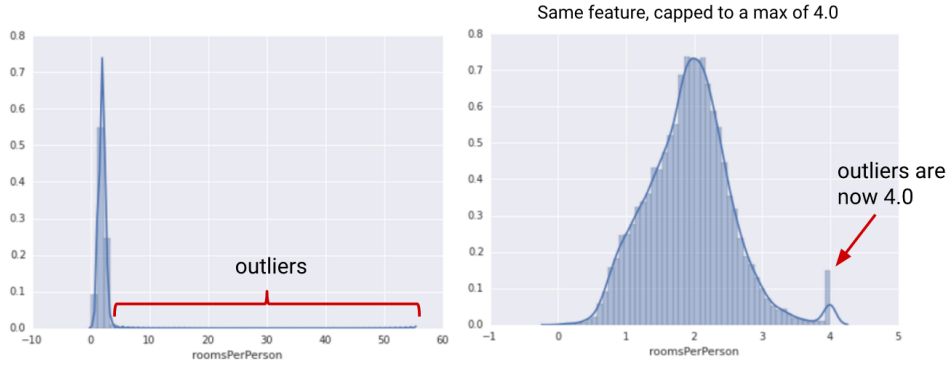
[Google Developers, 2021a]

Transformation des données

Valeurs numériques : Normalisation (Coupure)

$$x' = \begin{cases} \alpha & \text{si } x \geq \alpha \\ \beta & \text{si } x \leq \beta \\ x & \text{sinon} \end{cases}$$

- On utilise cette normalisation si
  - il y a des valeurs aberrantes extrêmes.
- Exemple
  - Nombre des chambres par personne
- numpy.clip



Exemple de coupure
[Google Developers, 2021a]

Transformation des données

Valeurs numériques : Normalisation (Mise à l'échelle min-max)

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

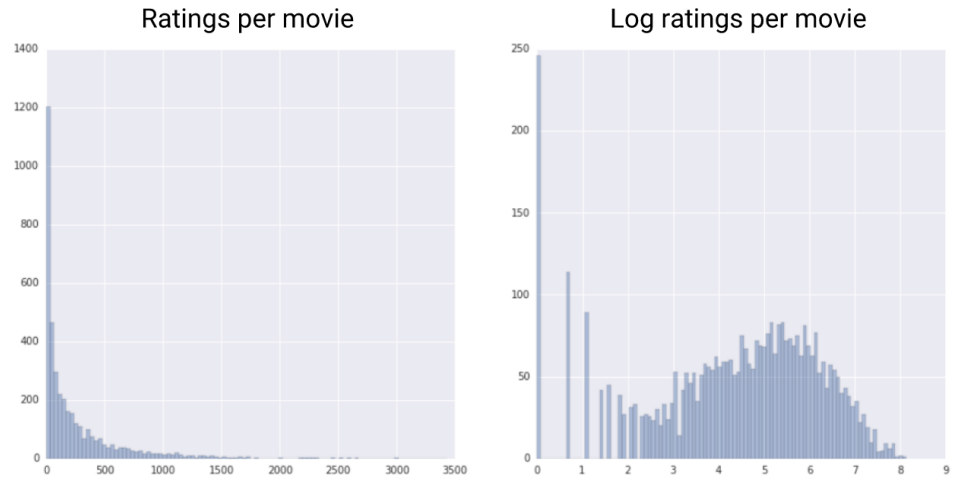
- On utilise cette normalisation si
  - On sait les limites inférieures et supérieures
  - Les valeurs sont presque uniformément réparties sur cette plage
- Exemple
  - Bon : les ages
  - Mauvais : les revenus (peu de personnes avec grand revenu)
- sklearn.preprocessing.MinMaxScaler

Transformation des données

Valeurs numériques : Normalisation (Mise à l'échelle log)

$$x' = \log(x)$$

- On utilise cette normalisation lorsque
  - la caractéristique conforme à la loi de puissance
- Exemple
  - Nombre des chambres par personne
- numpy.log



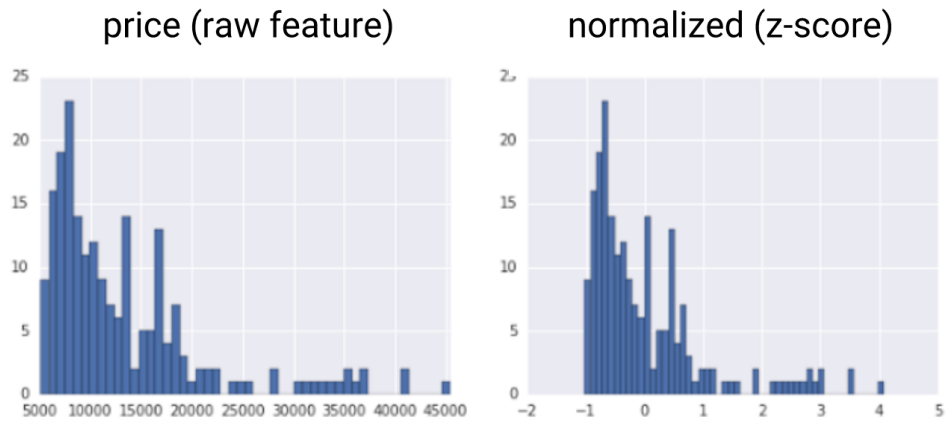
Exemple de normalisation log
[Google Developers, 2021a]

Transformation des données

Valeurs numériques : Normalisation (Z-score, standardisation)

$$x' = \frac{x - \mu}{\sigma}$$

- On utilise cette normalisation si
  - il n'y a pas trop de valeurs aberrantes.
  - on veut avoir une moyenne de 0 et déviation standard de 1
- Exemple
  - Nombre des chambres par personne
- sklearn.preprocessing.StandardScaler



Exemple de standardisation
[Google Developers, 2021a]

Transformation des données

Valeurs numériques : Génération de caractéristiques

- On utilise la génération de caractéristiques lorsque
  - on veut avoir des représentations complexes
- Génération polynomiale (les interactions)
  - $(X_1, X_2, \ldots) \longrightarrow (1, X_1, X_2, X_1X_2, X_1^2, X_2^2, \ldots)$
  - sklearn.preprocessing.PolynomialFeatures
- Auto-encodeurs épars
  - Un réseaux de neurones avec apprentissage non supervisé
  - Il vise à représenter les échantillon avec plus de caractéristiques (une représentation vectorielle avec plus de dimensionnalité)

Transformation des données

Valeurs nominales : Encodage

- Encodage ordinal
  - l'API utilisé pour l'entraînement n'accepte pas des chaines de caractères
  - l'algorithme d'apprentissage ne considère pas les catégories comme ordonnées : Naïve Bayes multinomial
  - ou si l'ordre des catégories est important
  - sklearn.preprocessing.OrdinalEncoder
- Encodage One-Hot
  - Lorsqu'on veut donner la même chance aux différentes catégories
  - Chaque catégorie de la caractéristique sera encodée sous forme binaire dans une colonne à part
  - sklearn.preprocessing.OneHotEncoder

Transformation des données

Un peu d'humour

Section 3

Échantillonnage et fractionnement des données

Échantillonnage et fractionnement des données

Données déséquilibrées

Degré de déséquilibre	Proportion de classe minoritaire
léger	20-40% de données
modéré	1-20% de données
extrême	<1% de données

Table – Degré de déséquilibre [Google Developers, 2021a]

- https://github.com/scikit-learn-contrib/imbalanced-learn
- pip install -U imbalanced-learn
- conda install -c conda-forge imbalanced-learn

Échantillonnage et fractionnement des données

Données déséquilibrées : Sous échantillonnage

- Supprimer des échantillons de la classe majoritaire (downsampling)
- Calibrer le modèle (upweithing)
- Suppression aléatoire
  - imblearn.under_sampling.RandomUnderSampler
- Centroids des clusters
  - Utiliser K-Means sur la classe majoritaire (avec K les nombre des échantillons voulus)
  - Prendre le centre de chaque cluster comme représentant
  - imblearn.under_sampling.ClusterCentroids
- Liens de Tomek
  - Détecter les points de la classe majoritaire les plus proches aux points de la classe minoritaire
  - Supprimer ces points
  - imblearn.under_sampling.TomekLinks
- I'm too lazy to present more methods! :)

Échantillonnage et fractionnement des données

Données déséquilibrées : Sur-échantillonnage

- Ajouter des échantillons de la classe minoritaire (oversampling)
- Duplication aléatoire
  - imblearn.over_sampling.RandomOverSampler
- SMOTE (Synthetic Minority Over-sampling Technique)
  - Chercher les K voisins les plus proches de chaque point de la classe minoritaire
  - Définir un nouveau point entre ce point et un de ces voisins
  - imblearn.over_sampling.SMOTE
- ADASYN (Adaptive Synthetic Sampling)
  - Comme SMOTE
  - La méthode favorise les points dans des espaces non homogènes
  - imblearn.over_sampling.ADASYN
- Still lazy

Échantillonnage et fractionnement des données

Fractionnement des données

- Division des données
  - Entrainement : une majorité des échantillons (70-80%)
  - Test : une minorité des échantillons (30-20%)
  - Validation : une minorité des échantillons pour régler les hyper-paramètres (20% à partir de l'entrainement)
- Condition sur la division
  - Les données de test sont suffisantes pour avoir des résultats significatifs.
  - Les données de test sont représentatives. Il ne faut pas prendre un ensemble avec des caractéristiques différentes de celles des données d'entrainement.
  - Si le problème est une prédiction du futur à base du passé, il faut diviser le dataset tel que le test suit l'entrainement
  - Pour ne pas perdre les données d'entrainement, utiliser la validation croisée

Échantillonnage et fractionnement des données

Fractionnement des données : Données et apprentissage

- Sous-apprentissage (Underfitting)
  - Le modèle n'a pas pris de temps pour généraliser sur les données d'entrainement
  - Peu de données d'entrainement
  - Les données d'entrainement ne sont pas représentatives
- Sur-apprentissage (Overfitting)
  - Le modèle a appris un apprentissage "par cœur" des données
  - Il existe du bruit dans les données d'entrainement
  - Les données de test ne sont pas représentatives.

Échantillonnage et fractionnement des données

Fractionnement des données : Échantillonnage stratifié

- Division aléatoire des données
  - Les données de test peuvent ne pas contenir des classes
  - Les données de test peuvent ne pas être proportionnelles à celles d'entrainement
- Étapes
  - Séparer les données selon leurs classes
  - Prendre des échantillons aléatoirement de chaque classe pour former le dataset de test
  - On peut calculer les proportions originales de chaque classe dans les données et extraire des échantillons selon ses proportions

Échantillonnage et fractionnement des données

Fractionnement des données : Validation croisée

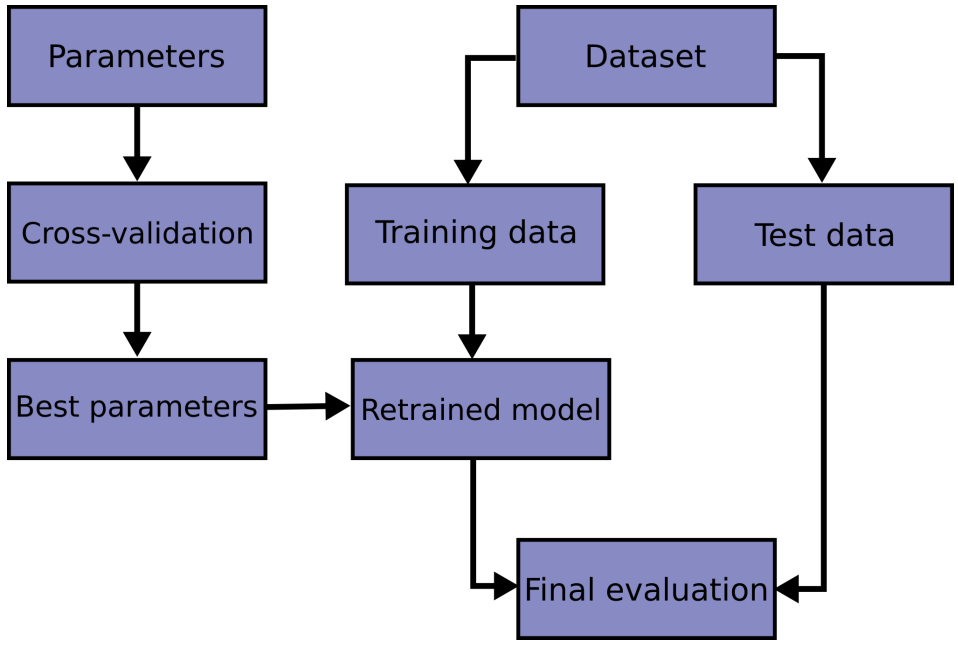


Figure – Workflow de l'apprentissage automatique avec validation croisée [scikit-learn developers, 2020]

Échantillonnage et fractionnement des données

Fractionnement des données : Validation croisée (K-Folds)

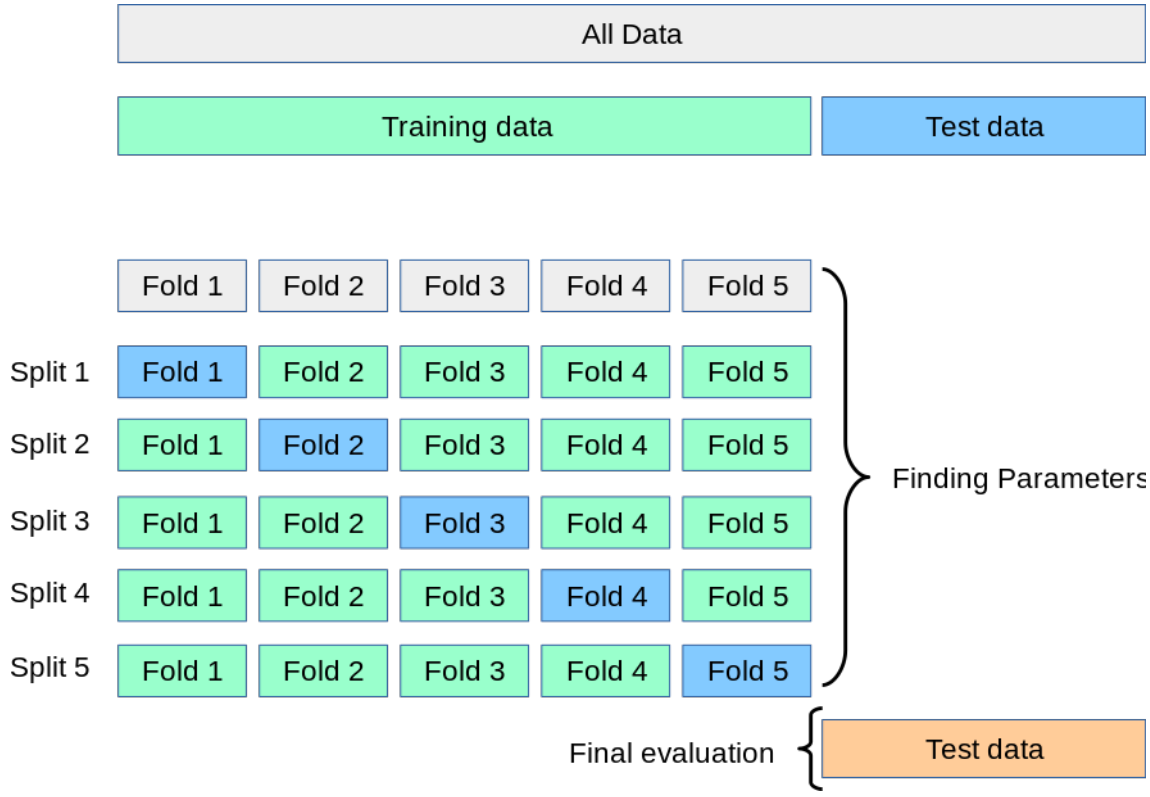
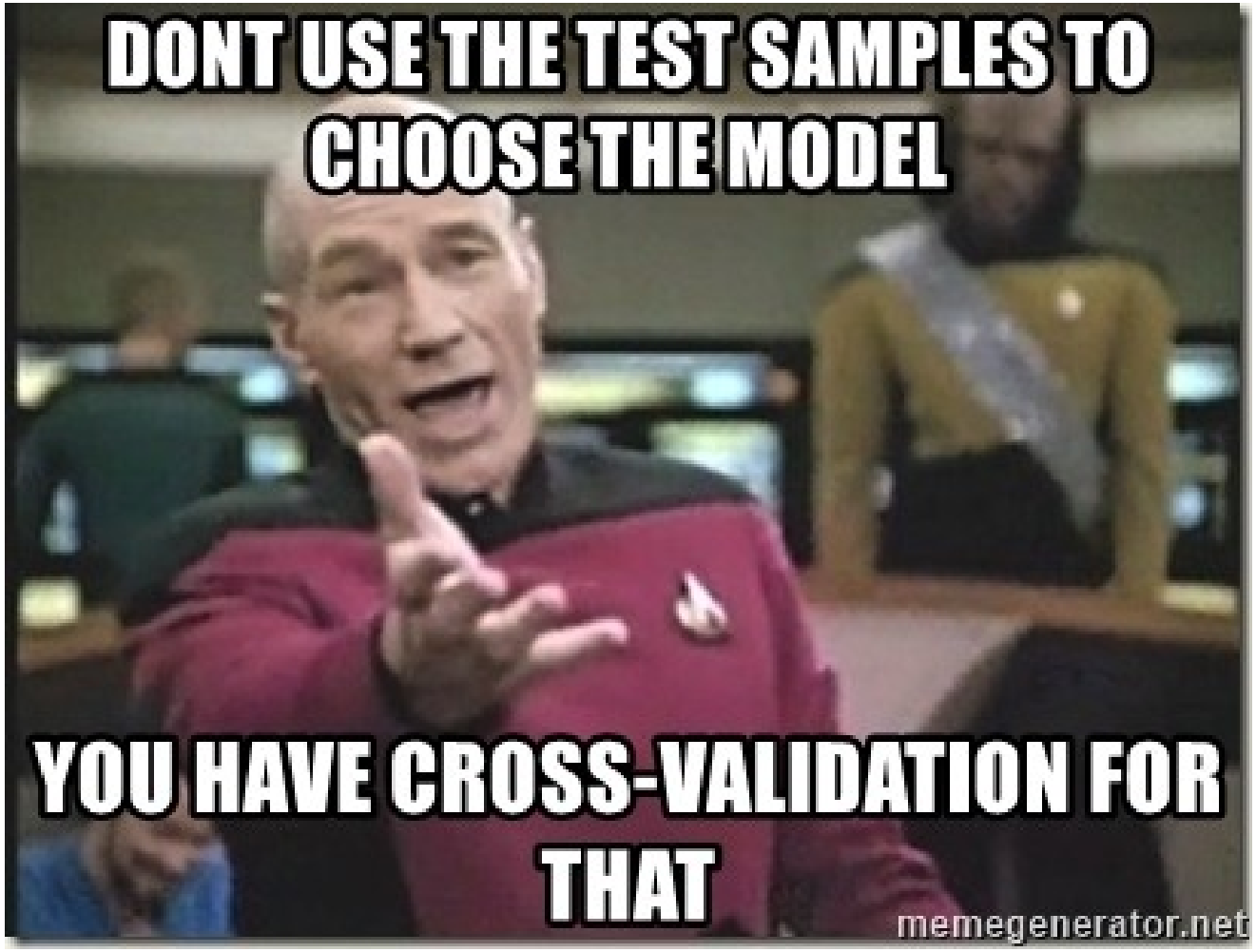


Figure – Illustration de la validation croisée K-Folds [scikit-learn developers, 2020]

Échantillonnage et fractionnement des données

Un peu d'humour



Section 4

Évaluation des modèles

Classement : Matrice de confusion

		Classes réelles	
		Positive	Négative
Prédiction	Positive	vrai positif (TP)	faux positif (FP)
	Négative	faux négatif (FN)	vrai négatif (TN)

- Vrai positif (True positive) : Le modèle prédit correctement la classe positive.
- Vrai négatif (True negative) : Le modèle prédit correctement la classe négative.
- Faux positif (False positive) : Le modèle prédit incorrectement la classe positive.
- Faux négatif (False negative) : Le modèle prédit incorrectement la classe négative.

Classement : Justesse (Accuracy)

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

		Classes réelles	
		indésirable	désirable
Prédiction	indésirable	0	1
	désirable	3	16

- $Accuracy = \frac{0+16}{1+1+3+16} = \frac{16}{20} = 80\%$
- La justesse n'est pas une bonne métrique si
  - les classes sont déséquilibrées
  - on veut avoir la performance du modèle sur la classe positive
- sklearn.metrics.accuracy_score

Classement : Rappel, Précision et F1 score

$$R = \frac{TP}{TP + FN} = \frac{\text{prédicitions positives justes}}{\text{échantillons positifs réels}}$$

- La capacité de trouver tous les échantillons positifs par le classificateur
- sklearn.metrics.recall_score

$$P = \frac{TP}{TP + FP} = \frac{\text{prédicitions positives justes}}{\text{prédictions positives}}$$

- La capacité de ne pas marquer des échantillons négatifs comme positifs par le classificateur
- sklearn.metrics.precision_score

$$F1 = \frac{2PR}{P + R}$$

- Moyenne harmonique entre P et R
- sklearn.metrics.f1_score

Classement : R, P et F1 (multi-classes)

- $\hat{y}$: l'ensemble des prédictions, y : l'ensemble des étiquettes justes
- L : l'ensemble des étiquettes (classes)
- S : l'ensemble des échantillons
- M : la métrique qui peut être R, P ou $F1$

Moyenne	Formule de calcul
micro (Précision)	$\frac{\sum_{l \in L} TP_l}{\sum_{l \in L} (TP_l + FP_l)}$
macro	$\frac{1}{\lvert L \rvert} \sum_{l \in L} M(y_l, \hat{y}_l)$
pondérée	$\frac{1}{\sum_{l \in L} \lvert \hat{y}_l \rvert} \sum_{l \in L} \lvert \hat{y}_l \rvert M(y_l, \hat{y}_l)$

Classement : Corrélation de matthews

$$CCM = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- Le coefficient peut être entre -1 et +1
  - -1 : les prédictions sont totalement irronnées.
  - 0 : la performance du modèle est comparable avec un système aléatoire (random).
  - +1 : les prédictions sont parfaites.
- sklearn.metrics.matthews_corrcoef

Régression : Erreur quadratique moyenne

$$MSE(\hat{y}, y) = \mathbb{E}(y - \hat{y})^2 = \frac{1}{|S|}\sum_{i=1}^{|S|}(y_i - \hat{y}_i)^2$$

- Mean Squared Error
- Punir le modèle lorsqu'il y a des grandes erreurs (magnifier les grandes erreurs)
- En réalité, calcule l'erreur carrée
- sklearn.metrics.mean_squared_error
- Pour avoir l'erreur, on peut appliquer une racine carrée
- $RMSE(\hat{y}, y) = \sqrt{MSE(\hat{y}, y)}$
- Root Mean Squared Error

Régression : Erreur absolue moyenne

$$MAE(\hat{y}, y) = \mathbb{E}|y - \hat{y}| = \frac{1}{|S|}\sum_{i=1}^{|S|}|y_i - \hat{y}_i|$$

- Mean Absolute Error
- Considère les petites erreurs et les grandes erreurs de la même magnitude
- sklearn.metrics.mean_absolute_error

Regroupement : Indice de Rand

$$RI(\hat{y}, y) = \frac{|\hat{y} \cap y|}{|y|}$$

- Il faut avoir un corpus annoté
- sklearn.metrics.rand_score

Regroupement : Silhouette

- C l'ensemble des clusters
- a_i est la distance moyenne du point i avec les points du même cluster
- b_i est la distance moyenne du point i avec les points du cluster le plus proche
- sklearn.metrics.silhouette_score

$$a_i = \frac{1}{|C_k| - 1} \sum_{j \in C_k, i \neq j} d(i,j) / i \in C_k$$

$$b_i = \min_{i \notin C_k} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

Évaluation des modèles

Un peu d'humour

Section 5

Bibliographie

Bibliographie

Google Developers (2021a).

Data preparation and feature engineering in ml.
Cours en ligne.
https://developers.google.com/machine-learning/data-prep [visité : 05 Avril 2021].

Google Developers (2021b).

Machine learning crash course.
Cours en ligne.
https://developers.google.com/machine-learning/crash-course [visité : 05 Avril 2021].

scikit-learn developers (2020).

scikit-learn api reference.
Documentation.
https://scikit-learn.org/stable/modules/classes.html [visité : 05 Avril 2021].

The imbalanced-learn developers (2021).

imbalanced-learn documentation.
Documentation.
https://imbalanced-learn.org/stable/ [visité : 05 Avril 2021].