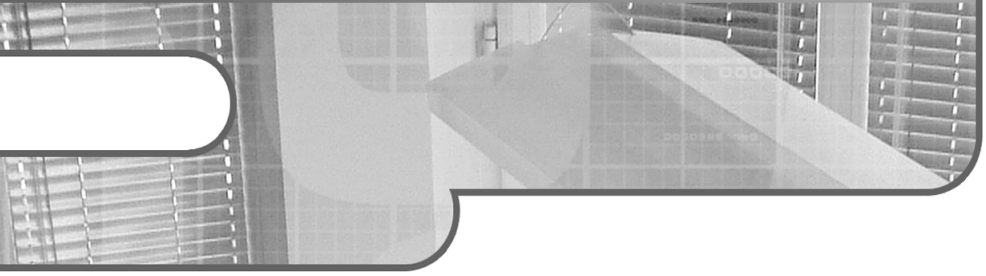



# Chapitre 5
# Préparation des données

## 1. La phase de Data Preparation

Dans la méthode CRISP-DM, la phase de Data Preparation permet de passer des données brutes, telles qu'extraites des sources de données, à des données utilisables par les différents algorithmes de Machine Learning.

Cette préparation est nécessaire pour deux raisons principales :

– La majorité des algorithmes ont des contraintes sur le format des données en entrée. Cela peut concerner leur type, par exemple uniquement des variables numériques, ou des contraintes sur leur format, comme des réels entre 0 et 1.

– Préparer les données permet de grandement améliorer les résultats des algorithmes, en extrayant ou en créant des colonnes plus adaptées au problème.

Cette phase doit être fortement documentée. En effet, il est vital de savoir exactement les choix qui ont été faits ainsi que les raisons qui les ont motivés. Cela permet de pouvoir valider ces choix d'un point de vue métier avant une potentielle mise en production des modèles, et de s'assurer que les résultats sont cohérents.

C'est aussi pendant la préparation que le choix de limiter les données utilisées pour la suite du processus est fait. Là encore, toutes les décisions prises doivent être documentées.

Remarque

*Cette phase est potentiellement très chronophage car elle peut représenter jusqu'à 50 % de la durée d'un projet.*

## 2. Limiter les données

Toutes les données brutes ne seront pas forcément utilisées pour la suite du processus. Pour des raisons d'optimisation du travail effectué, elles doivent être éliminées dès le début de la phase de préparation des données.

Il est ainsi possible d'éliminer des lignes, dites enregistrements, ou bien d'éliminer des colonnes, dites caractéristiques.

Voici une liste des principales raisons d'éliminer des enregistrements :

– Les lignes ne correspondent pas aux cas à traiter, car ce sont des cas trop particuliers.
– Elles contiennent des erreurs comme des âges négatifs.
– Trop de données sont manquantes et leur intérêt est donc moindre.

Pour les caractéristiques, il peut être judicieux d'éliminer les colonnes pour les raisons suivantes :

– Elles n'ont pas de rapport avec le domaine.
– Elles ne sont pas exploitables en l'état, comme des noms de personnes.
– Elles sont trop incomplètes et n'apportent donc qu'une information fortement partielle.
– Elles sont trop uniformes, comme une variable avec une seule valeur possible.
– Il s'agit d'un identifiant unique.
– Etc.

Remarque

*Attention : toute donnée supprimée aura forcément un impact sur le modèle créé. En éliminant certains cas, le modèle ne pourra pas faire des inférences correctes sur ceux-ci. Si vous éliminez par exemple les habitations de plus de 200 m² dans le dataset Boston, votre modèle ne saura pas prédire avec précision les prix des appartements de plus de 200 m², bien qu'un résultat soit toujours fourni par le modèle. Vous aurez donc à charge de créer un test en amont pour ne pas appeler le modèle quand vous sortez des bornes sur lesquelles vous l'avez entraîné. La documentation de cette phase est primordiale car elle peut avoir des impacts importants lors de la mise en production.*

## 2.1 Supprimer des colonnes

Avec Pandas, il y a deux grandes façons d'éliminer des colonnes : soit en précisant la liste des colonnes à garder, soit en indiquant directement le nom des colonnes à supprimer.

Pour supprimer des variables de manière explicite, il faut utiliser la fonction `drop` qui prend en paramètre la liste des noms de colonnes à supprimer et renvoie un nouveau `DataFrame`.

Le dataset Iris contient quatre variables explicatives qui sont les largeurs et longueurs des pétales et sépales de la fleur. Supprimer une de ces variables se fait par la ligne suivante :

```
new_df = iris_df.drop(columns=['sepal_length'])
```

Lorsque le nombre de colonnes à supprimer est grand comparé au nombre de colonnes à garder, il peut être plus pratique d'indiquer les caractéristiques à conserver.

Pour cela, il faut créer un nouveau `DataFrame` en extrayant les colonnes intéressantes. La ligne suivante permet de ne garder que la longueur des pétales et la classe dans le dataset Iris :

```
new_df = iris_df[['petal_length', 'class']]
```

## 2.2 Supprimer des enregistrements

Il est possible là encore de choisir de ne garder que certains enregistrements ou au contraire d'indiquer exactement quels enregistrements garder.

**Remarque**

*Il est tout à fait possible de le faire en indiquant les index et/ou les numéros des lignes. Cette solution est cependant à bannir, car un traitement rajouté avant la suppression peut complètement modifier le résultat obtenu. De plus, il est plus simple de documenter et de comprendre la suppression de lignes avec des âges négatifs que la suppression des lignes 5, 8 et 23 d'un dataset.*

Il est possible de supprimer les lignes contenant des données manquantes. C'est la fonction `dropna` qui sera utilisée. Celle-ci demande comme paramètres l'axe (`axis`, 0 pour éliminer des lignes, 1 pour des colonnes) et une méthode (`how`, qui vaut `'any'` ou `'all'`). Dans le cas de `any`, toute ligne/colonne contenant au moins une valeur nulle sera supprimée, alors que pour `all`, il faut que toutes les valeurs soient manquantes pour que la suppression se fasse. Il est aussi possible de préciser un sous-ensemble des colonnes à utiliser grâce à `subset`.

Dans le cas du dataset Titanic, il est donc possible de supprimer toutes les lignes où l'âge est vide grâce à la ligne suivante :

```
new_df = titanic_df.dropna(axis=0, subset=['Age'])
```

Il est aussi possible de ne garder que les lignes qui correspondent à une condition. C'est ainsi que, pour ne garder que les individus mineurs (au sens américain) dans le dataset Titanic, il faut utiliser la ligne suivante :

```
new_df = titanic_df[titanic_df['Age'] <= 21]
```

Les opérateurs de comparaison sont utilisables sur toutes les colonnes, en testant l'égalité comme la comparaison. Pour garder uniquement les individus de genre féminin, il est donc possible de faire :

```
new_df = titanic_df[titanic_df['Sex'] == 'female']
```

Enfin, plusieurs conditions peuvent se cumuler, par exemple pour ne garder que les jeunes filles (femmes et mineures) :

```
new_df = titanic_df[(titanic_df['Sex'] == 'female') &
(titanic_df['Age'] <= 21)]
```

## 3. Séparer les datasets

Avant d'aller plus loin, il est primordial d'avoir au moins deux datasets :

– Un dataset d'entraînement, qui servira à créer le modèle.

– Un dataset de test, pour tester le modèle.

Le dataset de test ne devra plus être utilisé jusqu'à la fin du processus complet, et surtout pas pour modifier les modèles (qui seraient alors fortement biaisés). Il ne doit pas non plus servir à choisir quelles sont les meilleures préparations des données.

Remarque

*En anglais, analyser trop finement les données sans avoir extrait le dataset de test s'appelle le data snooping. Normalement, cette séparation devrait même avoir lieu avant la phase d'analyse des données pour ne pas inclure trop de biais statistiques dans la suite du processus.*

Lors du processus de modélisation, le dataset d'entraînement sera de nouveau séparé entre apprentissage et validation, la validation permettant de choisir les hyperparamètres des différents modèles.

### 3.1 Proportion Entraînement/Test

Pendant de nombreuses années, les textes de référence indiquaient qu'il fallait un ratio de 80/20, soit 80 % des données pour l'entraînement et 20 % pour le test. Cela est toujours vrai lorsque le nombre d'échantillons est faible, mais aujourd'hui, avec l'avènement du Big Data, c'est de moins en moins le cas.

En effet, sur un dataset de 150 valeurs comme Iris, il semble important d'avoir au moins 30 enregistrements pour tester les modèles. Cela représente environ dix lignes par classe.

Sur un dataset d'un million de données, il n'est cependant pas nécessaire d'avoir 200000 échantillons de test.

La proportion d'enregistrements dans le dataset de test doit donc être de 20 % pour les petits datasets. Cette proportion baissera d'autant que le dataset contient beaucoup d'enregistrements.

Il n'y a cependant pas de règles, car plus la sortie est complexe et plus le modèle devra être testé attentivement sur de nombreux cas.

Lorsque la proportion relative de chaque dataset est choisie, il faut alors séparer le dataset initial.

## 3.2 Séparation aléatoire

La méthode la plus classique consiste à garder les premiers X % pour un dataset et le reste pour le deuxième dataset.

Avec Pandas, il est possible d'utiliser les fonctions `head` et `tail` pour récupérer les premières ou dernières lignes d'un dataset. Le nombre de lignes peut être précédé d'un signe '-' voulant dire 'sauf', ou encore 'en partant de l'opposé'.

```
# calcul de la limite
TRAIN_RATIO = 0.8
nb_train = int(TRAIN_RATIO * titanic_df.shape[0])

# TRAIN : les nb_train premières lignes
train_titanic = titanic_df.head(nb_train)

# TEST : tout sauf les nb_train premières lignes
test_titanic = titanic_df.tail(-nb_train)
```

Ce n'est cependant pas conseillé car l'ordre du dataset initial est conservé. Il arrive souvent que les données soient triées et cette procédure ne permet pas d'avoir une bonne répartition dans les deux datasets.

Une meilleure solution consiste donc à mélanger le dataset avant de sélectionner les lignes. La librairie Pandas offre une solution deux-en-un (mélange et sélection) avec la fonction `sample`.