



Expert

Machine Learning

Implémentation en Python avec Scikit-learn

En téléchargement

 code source et datasets

 template CRISP

 + QUIZ

Version en ligne
OFFERTE !
pendant 1 an

Virginie MATHIVET



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EIMLPYTSL** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 1
Présentation du livre

- 1. Pourquoi un livre sur le Machine Learning ? 13
- 2. Python et Scikit-learn : les raisons du choix 14
- 3. À qui s'adresse ce livre ? 15
- 4. Organisation du livre et éléments en téléchargement 16
- 5. Datasets utilisés dans ce livre 17
 - 5.1 Iris de Fisher, 1936 17
 - 5.2 Titanic, 1994 18
 - 5.3 Boston, 1978 19

Chapitre 2
Le Machine Learning : vue d'ensemble

- 1. Un peu de vocabulaire 21
- 2. Les métiers de la data 24
- 3. La croissance du Machine Learning 27
- 4. Formes d'apprentissage et tâches de ML 28
 - 4.1 Apprentissage supervisé 29
 - 4.1.1 Classification 29
 - 4.1.2 Classification : le cas des images 30
 - 4.1.3 Régression 31
 - 4.1.4 Prévision 31
 - 4.2 Apprentissage non supervisé 32
 - 4.2.1 Clustering 32
 - 4.2.2 Réduction de dimensions 32

2 _____ Machine Learning

Implémentation en Python avec Scikit-learn

4.2.3	Système de recommandations	33
4.2.4	Associations	33
4.3	Apprentissage par renforcement	34
4.3.1	Comportements	35
4.3.2	Jeux et stratégies	35
4.4	Apprentissage semi-supervisé	36
4.5	Synthèse des différentes formes d'apprentissage et tâches	37
5.	Méthodologie CRISP-DM	37
5.1	Vue d'ensemble	38
5.2	Business Understanding	39
5.3	Data Understanding	41
5.3.1	Carte d'identité du dataset	41
5.3.2	Description des champs	42
5.3.3	Statistiques descriptives	42
5.4	Data Preparation	44
5.5	Modeling	45
5.6	Evaluation	46
5.7	Deployment	47

Chapitre 3

La pile technologique en Python

1.	Les outils de la Data Science	49
1.1	Les outils intégrés	50
1.2	L'auto ML	50
1.3	Les outils de développement	51
2.	Langage Python	51
2.1	Présentation	51
2.2	Brève présentation de R	52
2.3	Python ou R ?	53
2.4	Python 2 vs Python 3	53

- 3. Jupyter 54
 - 3.1 Caractéristiques de Jupyter 54
 - 3.2 Avantages de Jupyter pour la Data Science 58
 - 3.3 Installation et utilisation de Jupyter 59
- 4. Bibliothèques de Machine Learning 61
 - 4.1 NumPy 61
 - 4.2 Pandas 64
 - 4.3 Matplotlib 66
 - 4.4 Scikit-learn 69
- 5. Bibliothèques de Deep Learning 70

Chapitre 4
Chargement et analyse des données

- 1. La phase de Data Understanding 73
- 2. Chargement des données 74
- 3. Création de la carte d'identité du dataset 76
- 4. Description des champs 77
 - 4.1 Gestion des types 78
 - 4.2 Détection des données manquantes 81
- 5. Statistiques descriptives sur les champs 82
 - 5.1 Types de données 83
 - 5.2 Analyse des données numériques 83
 - 5.3 Graphiques sur les données numériques 86
 - 5.3.1 Histogramme 86
 - 5.3.2 Nuage de points 89
 - 5.3.3 Boîtes à moustaches 91
 - 5.4 Analyse sur les données catégorielles 93
 - 5.5 Graphiques sur les données catégorielles 95
 - 5.6 Autres données 98

4 _____ **Machine Learning**

Implémentation en Python avec Scikit-learn

5.7	Analyse croisée des données.	99
5.7.1	Entre des variables numériques	99
5.7.2	Entre des variables numériques et une variable catégorielle.	100
5.7.3	Entre des variables catégorielles.	101

Chapitre 5 **Préparation des données**

1.	La phase de Data Preparation	103
2.	Limiter les données.	104
2.1	Supprimer des colonnes	105
2.2	Supprimer des enregistrements	106
3.	Séparer les datasets.	107
3.1	Proportion Entraînement/Test	107
3.2	Séparation aléatoire	108
3.3	Séparation stratifiée	110
4.	Traiter les données manquantes	111
5.	Préparer les attributs numériques	113
5.1	Validation des données	113
5.1.1	Validation sémantique des données	113
5.1.2	Validation statistique des données	114
5.2	Feature engineering	116
5.3	Discrétisation	117
5.3.1	Intervalles égaux.	118
5.3.2	Répartition par quantile.	119
5.3.3	Répartition manuelle	119
5.4	Normalisation.	120
5.4.1	Normalisation min-max.	121
5.4.2	Normalisation standard.	122
5.4.3	Normalisation robuste.	122
5.4.4	Comparaison.	123

- 6. Préparer les catégories 126
 - 6.1 Validation des données 127
 - 6.2 Modification des catégories 128
 - 6.2.1 Ordonner ou réordonner des catégories 128
 - 6.2.2 Modifier la liste des catégories 128
 - 6.3 Quantification 131
- 7. Les données particulières 134
 - 7.1 Préparer les dates 134
 - 7.1.1 Le format datetime64 135
 - 7.1.2 Extraire des composantes 136
 - 7.1.3 Gérer les écarts 137
 - 7.2 Préparer les chaînes de caractères 138
 - 7.2.1 Préparer les chaînes 139
 - 7.2.2 Effectuer une recherche dans les chaînes 140
 - 7.2.3 Extraire des sous-chaînes 141
 - 7.2.4 Autres méthodes 142
- 8. Automatiser la préparation 143
 - 8.1 Création de pipelines de traitement 143
 - 8.2 Paramètres des opérations et code Pandas 144
 - 8.3 Pipelines avec Scikit-learn 145
 - 8.3.1 Création d'un Transformer 145
 - 8.3.2 Utilisation des Transformer 146
 - 8.3.3 Inconvénients de Scikit-learn 148
 - 8.4 Autres possibilités 149

6 **Machine Learning**

Implémentation en Python avec Scikit-learn

Chapitre 6

Modélisation et évaluation

1. Phase de modélisation	151
2. Création d'un ensemble de validation	152
3. Préparation des datasets	154
3.1 Dataset Iris	154
3.2 Dataset Titanic	155
3.3 Dataset Boston	157
4. Création des modèles	159
4.1 Processus itératif	159
4.2 Création d'un modèle en Scikit-learn	160
4.3 Évaluation d'un modèle	161
4.4 Validation croisée	161
4.5 Sauvegarde et chargement d'un modèle	162
5. Amélioration des modèles (fine-tuning)	164
5.1 Optimisation des hyperparamètres	164
5.2 Application en Scikit-learn	165
5.3 Sur- et sous-apprentissage	167
6. Méthodes ensemblistes	169
6.1 Bagging	169
6.2 Boosting	171
6.3 Stacking	172

Chapitre 7

Algorithmes de classification

1. La tâche de classification	175
1.1 Définition	175
1.2 Exemples de cas pratiques	176
1.3 Préparation spécifique des données	177

- 2. Évaluation des modèles 177
 - 2.1 Matrices de confusion 179
 - 2.1.1 Cas de la classification binaire 179
 - 2.1.2 Cas de la classification multiclasse 181
 - 2.2 Indicateurs dérivés de la matrice de confusion 182
 - 2.2.1 Accuracy 183
 - 2.2.2 Rappel et précision 184
 - 2.2.3 F1-score 187
 - 2.2.4 Sensibilité et spécificité 188
 - 2.3 La courbe ROC et l'AUC 190
 - 2.3.1 Prédiction et probabilité 190
 - 2.3.2 Taux de vrais et faux positifs 191
 - 2.3.3 Courbe ROC 191
 - 2.3.4 Aire sous la courbe (AUC) 194
 - 2.4 Choix des indicateurs d'évaluation 195
- 3. Les arbres de décision et algorithmes dérivés 195
 - 3.1 Arbres de décision 195
 - 3.1.1 Sortie de l'arbre 197
 - 3.1.2 Choix du point de coupure 197
 - 3.1.3 Critères d'arrêt 198
 - 3.1.4 Exploitation de l'arbre 200
 - 3.2 Random Forests 203
 - 3.3 XGBoost (eXtreme Gradient Boosting) 205
- 4. K-Nearest Neighbors 206
- 5. Logistic Regression 209
 - 5.1 Régression logistique binaire 209
 - 5.2 Régression logistique polytomique 211
 - 5.3 Application avec Scikit-learn 211
- 6. Naive Bayes 214
 - 6.1 Principe général 214
 - 6.2 Calcul des différentes probabilités 215
 - 6.3 Application avec Scikit-learn 216

8 **Machine Learning**

Implémentation en Python avec Scikit-learn

7.	Support Vector Machine	218
7.1	Présentation générale	218
7.1.1	Marge et support vector	219
7.1.2	Kernels	220
7.1.3	Avantages	221
7.2	Application avec Scikit-learn	222

Chapitre 8

Algorithmes de régression

1.	La tâche de régression	225
1.1	Définition	225
1.2	Exemples de cas pratiques	226
1.3	Préparation spécifique des données	226
2.	Entraînement et évaluation des modèles	227
2.1	Notion d'erreurs	229
2.2	Indicateurs dérivés de la mesure d'erreurs	230
2.2.1	Erreur absolue moyenne	230
2.2.2	Erreur quadratique moyenne	231
2.2.3	Racine de l'erreur quadratique moyenne	232
2.2.4	Coefficient de détermination et variance expliquée	232
2.2.5	Autres indicateurs	234
2.3	Choix des indicateurs d'évaluation	235
3.	Utilisation des algorithmes de classification	235
3.1	Principe général	235
3.2	Arbres de décision et algorithmes dérivés	236
3.2.1	Arbres de décision	236
3.2.2	Random Forest	238
3.2.3	XGBoost	239
3.3	K-plus proches voisins (KNN)	239
3.4	SVM	240

- 4. Régression linéaire et variantes 241
 - 4.1 Régression linéaire 241
 - 4.2 Application dans Scikit-learn 242
 - 4.3 Problème de la colinéarité 245
 - 4.4 Ridge Regression 246
 - 4.5 Régression Lasso 248
- 5. Régression polynomiale 250
 - 5.1 Principe 250
 - 5.2 Régression polynomiale et Scikit-learn 251
- 6. Cas particulier de la prédiction 252
 - 6.1 Prédiction et séries temporelles 252
 - 6.2 Préparation des données 255
 - 6.3 Application en Scikit-learn 257
 - 6.4 Utilisation de modèles spécifiques 262
 - 6.4.1 Limites de l'approche en régression linéaire 262
 - 6.4.2 Algorithmes dédiés aux séries temporelles 264

Chapitre 9
Algorithmes d'apprentissage non supervisés

- 1. Les tâches en apprentissage non supervisé 269
- 2. Clustering 270
 - 2.1 Définition 270
 - 2.2 Exemples de cas pratiques 271
 - 2.3 Algorithmes basés sur les distances 272
 - 2.3.1 Principe de l'algorithme K-Means 272
 - 2.3.2 Implémentation avec Scikit-learn 273
 - 2.3.3 Variantes de l'algorithme K-Means 277
 - 2.4 Algorithmes basés sur la densité 278
 - 2.4.1 Principe général 278
 - 2.4.2 Implémentation de DBSCAN en Scikit-learn 279
 - 2.4.3 Variante de DBSCAN : OPTICS 280

3.	Réduction des dimensions	280
3.1	Définition	280
3.2	Exemples de cas pratiques	281
3.3	Détection des axes principaux.	282
3.4	Création de nouveaux axes	283
3.4.1	Principal Component Analysis (PCA).	283
3.4.2	Linear Discriminant Analysis (LDA).	286
4.	Systèmes de recommandation.	288
4.1	Définition	288
4.2	Principales approches	289
4.2.1	Modèles basés sur la popularité	289
4.2.2	Modèles basés sur le contenu (content-based filtering)	289
4.2.3	Modèles basés sur les autres utilisateurs.	290
4.2.4	Méthodes hybrides	291
5.	Association	291
5.1	Définition	291
5.2	Évaluation des algorithmes	292
5.2.1	Le support	293
5.2.2	L'indice de confiance.	293
5.2.3	Le lift	294
5.3	Algorithme "APriori".	295
5.3.1	Étape 1 : réalisation des comptages des ensembles	295
5.3.2	Étape 2 : création et test des règles	296

Chapitre 10

Évaluation et déploiement

1.	Phase d'évaluation	297
1.1	Principe global.	297
1.2	Évaluation métier des résultats	298
1.3	Revue du processus.	298
1.4	Étapes suivantes	299

- 2. Phase de déploiement 300
 - 2.1 Planification du déploiement 300
 - 2.2 Monitoring et maintenance 301
 - 2.3 Rapport final et documentation 302
- 3. Déploiement et MLOps 302
 - 3.1 Retours sur le DevOps 302
 - 3.2 Apparition du MLOps 303
 - 3.3 Tâches couvertes par le MLOps. 304
 - 3.4 Critères de choix 305

Conclusion

- 1. Le Machine Learning, une compétence-clé 309
- 2. Mener un projet jusqu'au bout 310
- 3. Au-delà de la méthodologie 312
- 4. Expérimentation et expérience 312
- 5. Pour aller plus loin 314

- Index 315

12 _____ **Machine Learning**

Implémentation en Python avec Scikit-learn

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`.