

Résolution d'un Problème de Machine Learning

Cycle de vie de la donnée

I. Résolution d'un problème : Machine Learning

Analyse de la problématique
Collecte des Données
Exploration Numérique et Visuelle des données
Préparation des données
Sélection de métrique
Sélection du modèle et son entraînement
Ajustement du modèle
Vérification du modèle
Solution
Mise en production et Déploiement

II. Cycle de vie de la donnée

Collecter
Stocker
Infrastructure
Modéliser
Analyser

III. Cycle de vie de la donnée à travers l'utilisation des bibliothèques Python suivantes :

- Pandas,
- NumPy,
- Matplotlib,
- Scikit-learn

- TensorFlow.

Chacune joue un rôle spécifique dans la collecte, la visualisation, la modélisation et l'évaluation des données.

1. Collecter/Préparer : Charger, nettoyer, structurer les données Pandas, NumPy
2. Visualiser Comprendre les patterns, corrélations : Matplotlib
3. Modéliser Créer un modèle prédictif ou descriptif : Scikit-learn, TensorFlow
4. Évaluer/Optimiser : Interpréter, affiner, valider les modèles Matplotlib, NumPy, Pandas

1. Collecter et préparer les données → Pandas + NumPy

- Pandas permet de charger des fichiers CSV, Excel, JSON, bases SQL...Nettoyage des données : gestion des valeurs manquantes, normalisation, jointures.
- NumPy transforme les données en matrices numériques exploitables (vecteurs, arrays).

Exemple :

```
import pandas as pd
```

```
import numpy as np
```

```
df = pd.read_csv("clients.csv")
```

```
X = df[["âge", "revenu"]].to_numpy()
```

2. Explorer et visualiser les données → Matplotlib

- Visualisation des distributions, corrélations, anomalies.
- Aide à comprendre la structure des données avant la modélisation.

Exemple :

```
import matplotlib.pyplot as plt

plt.scatter(df["âge"], df["revenu"])

plt.xlabel("Âge")

plt.ylabel("Revenu")

plt.title("Répartition des clients")

plt.show()
```

3. Modéliser les données

- **Modèle classique (machine learning) → Scikit-learn**

Exemple :

```
from sklearn.linear_model import LogisticRegression

import pandas as pd


df = pd.read_csv("clients.csv")

X = df[["âge", "revenu"]].to_numpy()

y = df["achat"] # par exemple une colonne binaire : 0 = non, 1 = oui


model = LogisticRegression()

model.fit(X, y)
```

- Modèle profond (deep learning) → TensorFlow

Exemple :

```
import tensorflow as tf

from sklearn.model_selection import train_test_split

import pandas as pd


df = pd.read_csv("clients.csv")

X = df[["âge", "revenu"]].to_numpy()

y = df["achat"] # Colonne cible binaire (0/1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(16, activation="relu",
input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(8, activation="relu"),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])

model.fit(X_train, y_train, epochs=10)
```

4. Évaluer, interpréter, améliorer

- Utilisation de Matplotlib pour visualiser les performances (précision, courbe ROC...)
- Pandas/NumPy pour analyser les erreurs, post-traiter les résultats.
- Réentraînement possible avec ajustement des paramètres ou nouvelles données.

Exemple :

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay  
ConfusionMatrixDisplay.from_estimator(model, X_test, y_test)  
plt.show()
```

IV. Résolution d'un problème : Machine Learning // Cycle de vie de la donnée :

1. Collecte

Analyse de la problématique

Comprendre le besoin métier ou scientifique :
Que cherche-t-on à prédire, classer, optimiser ? Quelle est la valeur ajoutée du projet ML ?

Exemple : Prédire si un patient est à risque de maladie cardiaque.

Collecte des données

Réunir les données nécessaires au projet :
Données internes (CRM, bases médicales), données publiques (open data), capteurs, etc.
Attention à la qualité, la pertinence, la légalité (RGPD) et l'éthique.

2. Stocker & Infrastructure

Stocker les données et les gérer efficacement

Utilisation d'un système de stockage : fichiers plats (CSV), bases de données (SQL/NoSQL), cloud (AWS, GCP, Azure).
Gestion des formats, des accès, et des mises à jour.

3. Modéliser

Exploration numérique et visuelle des données (EDA)

Comprendre les données avec des statistiques descriptives et des visualisations (boxplots, corrélations, histogrammes, etc.)
Identifier les erreurs, valeurs aberrantes, tendances.

Préparation des données

Nettoyage, normalisation, encodage, création de variables dérivées, gestion des valeurs manquantes.

Étape essentielle : 80% du temps en ML y est consacré.

Sélection de la métrique

Choisir une ou plusieurs métriques pour évaluer le modèle :

- Classification :
- Régression :
- Clustering :

Sélection du modèle et son entraînement

Choisir un ou plusieurs algorithmes adaptés (régression logistique, arbres, SVM, réseaux de neurones, etc.)

Entraîner le modèle avec les données préparées.

4. Analyser

Ajustement du modèle

Optimiser les hyperparamètres (grid search, random search, cross-validation), réduire le sur-apprentissage (overfitting).
Sélection de features, régularisation...

Vérification du modèle

Évaluer les performances sur un jeu de test indépendant.
Analyser les erreurs, biais, robustesse.
Peut nécessiter un retour aux étapes précédentes si les performances sont faibles.

Solution

Interpréter les résultats, traduire les prédictions en décisions concrètes.
Présenter les résultats aux parties prenantes (graphes, dashboards, rapports).

5. Mise en production et déploiement

Mise en production et déploiement

Intégrer le modèle dans un environnement opérationnel : via une API, dans une application web, sur un serveur, etc.
Inclut : surveillance du modèle, mises à jour, versioning, gestion de la dérive.