📚 Cours Complet XGBoost

Régression et Classification Expliquées

1. Introduction à XGBoost

🎯 Qu'est-ce que XGBoost ?

XGBoost signifie **eXtreme Gradient Boosting**. C'est un algorithme d'apprentissage automatique développé par Tianqi Chen en 2014, devenu l'algorithme le plus populaire pour les compétitions de Data Science.

🏆 Pourquoi XGBoost est si populaire ?

⚡ Performance Exceptionnelle

Gagne plus de 50% des compétitions Kaggle

🚀 Rapidité

Optimisé et parallélisé pour la vitesse

🛡️ Robustesse

Gère automatiquement les données manquantes

🎯 Flexibilité

Fonctionne pour régression ET classification

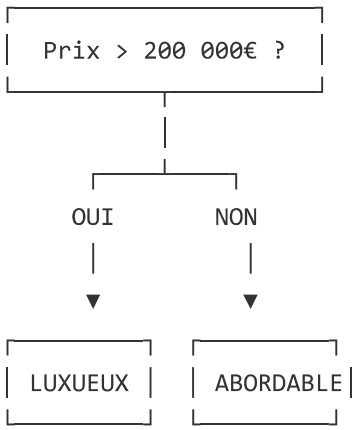
Applications Réelles

Secteur	Applications
🏦 Finance	Détection de fraudes, scoring crédit, prédiction de défaut
🏥 Santé	Diagnostic médical, prédiction de maladies
🛒 E-commerce	Prédiction de ventes, recommandations produits
🏠 Immobilier	Estimation de prix, évaluation de propriétés
📱 Marketing	Prédiction de churn, ciblage publicitaire

2. Concepts Fondamentaux

🌳 Les Arbres de Décision

Un arbre de décision est comme un questionnaire qui pose des questions pour arriver à une décision :

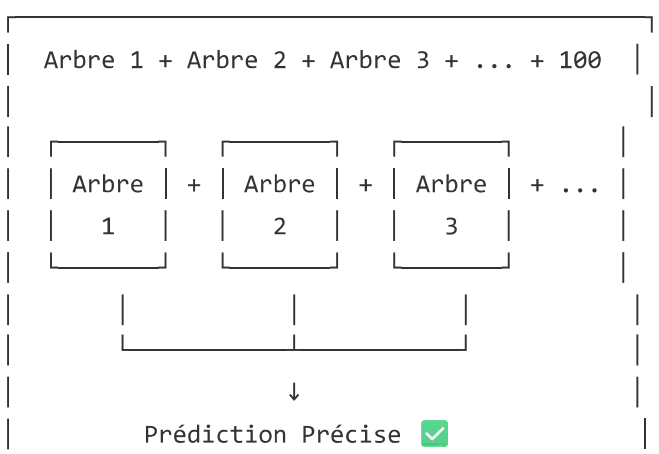


⚠️ Problème d'un Seul Arbre

- ❌ Peu précis
- ❌ Surapprentissage facile
- ❌ Instable

🚀 Le Boosting : La Solution

Le **Boosting** consiste à combiner plusieurs arbres faibles pour créer un modèle fort :



Processus d'Entraînement

```
ÉTAPE 1: Créer le premier arbre
         ↓
ÉTAPE 2: Calculer les erreurs
         ↓
ÉTAPE 3: Créer un nouvel arbre
         qui corrige ces erreurs
         ↓
ÉTAPE 4: Répéter jusqu'à convergence
         ↓
RÉSULTAT: Modèle puissant ✅
```

Exemple Concret

Objectif : Prédire un prix de 300 000€

Itération	Prédiction	Erreur	Action
Init	200 000€	-100k	Moyenne
Arbre 1	+60 000€	-40k	Correction
Arbre 2	+25 000€	-15k	Amélioration
Arbre 3	+10 000€	-5k	Affinement
Arbre 4	+5 000€	0k	✅ Parfait !

Résultat : 200k + 60k + 25k + 10k + 5k = **300 000€**

3. XGBoost pour la Régression

📊 Définition

Régression = Prédire une **valeur numérique continue**

Exemples :

- 💰 Prix d'une maison : 250 000€
- 🌡️ Température : 23.5°C
- 📈 Salaire : 45 000€/an
- ⏱️ Temps : 15.8 minutes

🏠 Cas Pratique : Prédiction de Prix Immobilier

Le Problème

Variables d'entrée :

- Surface (m^2)
- Nombre de chambres
- Âge du bien (années)
- Distance du centre (km)

À prédire : Prix en milliers d'euros

Exemple de Données

Surface	Chambres	Âge	Distance	Prix (k€)
50	1	30	15	**150**
90	2	15	8	**250**

130	4	5	3	**380**
150	5	2	2	**480**

Code Python

```
# ================================================
# IMPORTS
# ================================================
import xgboost as xgb
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# ================================================
# DONNÉES
# ================================================
data = {
    'surface': [50, 70, 90, 110, 130, 150],
    'chambres': [1, 2, 2, 3, 4, 5],
    'age': [30, 20, 15, 10, 5, 2],
    'distance': [15, 10, 8, 5, 3, 2],
    'prix': [150, 200, 250, 300, 380, 480]
}
df = pd.DataFrame(data)

# ================================================
# PRÉPARATION
# ================================================
X = df[['surface', 'chambres', 'age', 'distance']]
y = df['prix']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ================================================
# MODÈLE RÉGRESSION
# ================================================
```

```
model = xgb.XGBRegressor(
    objective='reg:squarederror',  # ← Pour RÉGRESSION
    n_estimators=100,
    learning_rate=0.1,
    max_depth=5,
    random_state=42
)

# ================================================
# ENTRAÎNEMENT
# ================================================
model.fit(X_train, y_train)

# ================================================
# PRÉDICTION
# ================================================
nouvelle_maison = [[100, 3, 10, 5]]
prix_predit = model.predict(nouvelle_maison)

print(f"Prix prédit : {prix_predit[0]:.2f}k€")

# ================================================
# ÉVALUATION
# ================================================
y_pred = model.predict(X_test)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"RMSE : {rmse:.2f}k€")
print(f"R²   : {r2:.3f}")
```

Métriques de Régression

RMSE

Root Mean Squared Error

R^2

Coefficient de Détermination

Qualité du modèle (0 à 1)

Mesure l'erreur moyenne en k€. Plus proche de 0 = meilleur.

Exemple : RMSE = 15k€ → erreur de ±15 000€

MAE

Mean Absolute Error

Erreur absolue moyenne, facile à interpréter.

Exemple : MAE = 10k€ → erreur de 10 000€

- 0.0 = Inutile ❌
- 0.8 = Très bon ✅
- 1.0 = Parfait 🎯

4. XGBoost pour la Classification

🎯 Définition

Classification = Prédire une **catégorie** (classe discrète)

Exemples :

- 📧 Email : Spam / Non-spam
- 🏥 Diagnostic : Malade / Sain
- 💳 Transaction : Fraude / Légitime
- 🚗 Type : Voiture / Camion

Types de Classification

Binaire (2 classes)

- Oui / Non
- Vrai / Faux
- 0 / 1

Multi-classes (>2 classes)

- Voiture / Camion / Moto
- A / B / C / D / E
- Rouge / Vert / Bleu

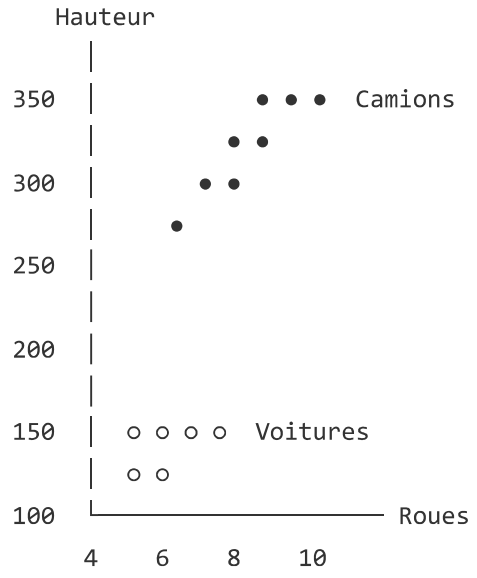
🚗 Cas Pratique : Classification de Véhicules

Le Problème

Variables d'entrée : Nombre de roues, Hauteur (cm)

À prédire : Type (Voiture ou Camion)

Visualisation des Données



Exemple de Données

Roues	Hauteur (cm)	Type

4	145	Voiture
6	280	Camion
4	150	Voiture
8	310	Camion

Code Python

```
# ================================================
# IMPORTS
# ================================================
import xgboost as xgb
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# ================================================
# DONNÉES
# ================================================
data = {
    'roues': [4, 4, 6, 4, 8, 4, 10, 4, 6, 4],
    'hauteur': [145, 150, 280, 155, 310, 148, 330, 152, 275, 160]
    'type': [0, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # 0=Voiture, 1=Camion
}
df = pd.DataFrame(data)

# ================================================
# PRÉPARATION
# ================================================
X = df[['roues', 'hauteur']]
y = df['type']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# ================================================
```

```
# MODÈLE CLASSIFICATION
# ================================================
model = xgb.XGBClassifier(
    objective='binary:logistic',  # ← Pour CLASSIFICATION
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)

# ================================================
# ENTRAÎNEMENT
# ================================================
model.fit(X_train, y_train)

# ================================================
# PRÉDICTION
# ================================================
nouveau_vehicule = [[6, 270]]
prediction = model.predict(nouveau_vehicule)
probabilites = model.predict_proba(nouveau_vehicule)

print(f"Type prédit : {'Camion' if prediction[0]==1 else 'Voiture
print(f"Probabilités :")
print(f"  Voiture : {probabilites[0][0]:.1%}")
print(f"  Camion  : {probabilites[0][1]:.1%}")

# ================================================
# ÉVALUATION
# ================================================
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision : {accuracy:.1%}")
```

Métriques de Classification

Accuracy

Precision

Pourcentage de prédictions correctes

95% = 95 bonnes sur 100

Parmi les prédictions positives, combien sont correctes ?

Recall

Parmi les vrais positifs, combien ont été détectés ?

F1-Score

Moyenne harmonique de Precision et Recall

Matrice de Confusion

```
                PRÉDIT
              Voiture  Camion
            ┌─────────┬─────────┐
RÉEL        │         │         │
Voiture     │   45    │    2    │  ← 45 OK, 2 erreurs
            ├─────────┼─────────┤
Camion      │    1    │   52    │  ← 52 OK, 1 erreur
            └─────────┴─────────┘
```

5. Hyperparamètres de XGBoost

Paramètres Principaux

Paramètre	Description	Valeurs
`objective`	Type de tâche	`reg:squarederror` (régression) `binary:logistic` (classification)