

Entrainement de modèle

Régression Linéaire

Entraînement d'un Modèle - RÉGRESSION

Définition : Prédire une valeur numérique (prix, température, salaire...)



Exemple : Prédire le prix d'un appartement

```
from sklearn.linear_model import LinearRegression
# Créer le modèle
model = LinearRegression() # Créer le modèle

# Entraîner sur les données de train
model.fit(X_train, y_train)
print("✅ Modèle entraîné !")

# Faire des prédictions sur le test
y_pred = model.predict(X_test)
print("✅ Modèle entraîné !")
```

Métriques de Régression

Comment savoir si mon modèle classe bien ?

MAE : L'erreur moyenne

(Mean Absolute Error)

**De combien je me
trompe
en moyenne ?**

*Plus le MAE est petit,
mieux c'est !*



Exemple

3 prédictions :

- Erreur 1 : 5 000€
- Erreur 2 : 15 000€
- Erreur 3 : 2 000€

MAE = 7 333€

RMSE : Attention aux grosses erreurs !

(Root Mean Squared Error)

**Comme MAE, mais punit
plus fort les grosses
erreurs**

*Utile quand les grosses
erreurs sont dangereuses*



Comparaison

Mêmes erreurs qu'avant :

MAE = 7 333€

RMSE = 9 200€

*RMSE > MAE car grosse
erreur de 15 000€*

R^2 : La note de mon modèle

(Coefficient de Détermination)



En français simple

C'est une note entre 0 et 1 (ou 0% et 100%)



Comment lire R^2 ?

- $R^2 = 0.90 \rightarrow$ Excellent ! Mon modèle explique 90%
- $R^2 = 0.70 \rightarrow$ Bon modèle (70%)
- $R^2 = 0.30 \rightarrow$ Mauvais modèle (30%)

Calculer les Métriques - RÉGRESSION

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Calculer les métriques
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# Afficher les résultats
print("📊 RÉSULTATS DU MODÈLE :")
print(f"MAE : {mae:.0f} MAD")
print(f"RMSE : {rmse:.0f} MAD")
print(f"R² : {r2:.2f}")

# Interpréter
print("\n💡 INTERPRÉTATION :")
print(f"→ Le modèle se trompe en moyenne de {mae:.0f} MAD")
print(f"→ Le modèle explique {r2*100:.0f}% de la variance du prix")

# C'est bon ?
if r2 > 0.8:
    print("✅ Très bon modèle !")
elif r2 > 0.6:
    print("⚠️ Modèle correct, peut être amélioré")
else:
    print("❌ Modèle faible, à retravailler")
```

En résumé

MAE

→ L'erreur moyenne (plus simple)

RMSE

→ Punit les grosses erreurs

R^2

→ La note globale (0 à 1)

Entrainement de modèle

Régression logistique

Entraînement d'un Modèle - CLASSIFICATION

Définition : Prédire une catégorie (Oui/Non, Chat/Chien, A/B/C...)



Exemple : Un email est SPAM ou PAS SPAM ?

```
from sklearn.linear_model import LogisticRegression

# Créer le modèle
model = LogisticRegression()
# Entraîner sur les données de train
model.fit(X_train, y_train)
print("✅ Modèle entraîné !")

# Faire des prédictions sur le test
y_pred = model.predict(X_test)
```

Métriques d'Évaluation

CLASSIFICATION

Comment savoir si mon modèle classe bien ?

Contexte : Détection de SPAM



Scénario

Un modèle détecte les emails spam sur
100 emails

Composition :

- 60 vrais spams
- 40 emails normaux



Résultats du modèle

		Prédictions →	
		SPAM	NORMAL
Réalité ↓	SPAM	48	12
	NORMAL	8	32

Métriques : Accuracy & Precision

Accuracy (Précision globale)

90%

Question :

Sur tous mes emails, combien j'ai bien classés ?

Calcul :

(Bonnes prédictions) / (Total d'emails)

$$90/100 = 0.90 = 90\%$$

Precision

85%

Question :

Parmi les emails que j'ai prédits comme SPAM, combien sont vraiment des spams ?

Calcul :

$$42/60 = 0.85 = 85\%$$

💡 Haute précision = peu de faux positifs

Métriques : Recall & F1-Score

Recall (Sensibilité)

80%

Question :

Parmi tous les vrais SPAM, combien j'en ai détectés ?

Calcul :

$$48/60 = 0.80 = 80\%$$

💡 *Haut recall = peu de faux négatifs*

F1-Score

82%

Question :

Quel est l'équilibre entre Precision et Recall ?

Calcul :

$$2 \times (P \times R) / (P + R)$$
$$2 \times (0.85 \times 0.80) / 1.65 = 0.82$$

💡 *Équilibre entre les deux métriques*

Calculer les Métriques - CLASSIFICATION

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix

# Calculer les métriques
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Afficher les résultats
print("📊 RÉSULTATS DU MODÈLE :")
print(f"Accuracy : {accuracy:.2f} ({accuracy*100:.0f}%)")
print(f"Precision : {precision:.2f}")
print(f"Recall : {recall:.2f}")
print(f"F1-Score : {f1:.2f}")

# Rapport détaillé
print("\n📄 RAPPORT DÉTAILLÉ :")
print(classification_report(y_test, y_pred))

# Matrice de confusion
print("\n🔍 MATRICE DE CONFUSION :")
print(confusion_matrix(y_test, y_pred))

# Interpréter
if accuracy > 0.9:
    print("\n✅ Excellent modèle !")
elif accuracy > 0.75:
    print("\n⚠️ Bon modèle, peut être amélioré")
else:
    print("\n❌ Modèle faible, à retravailler")
```

Résumé

✓ Accuracy

Performance globale du modèle

✓ Precision

Fiabilité des prédictions positives

✓ Recall

Capacité à détecter tous les positifs

✓ F1-Score

Équilibre entre Precision et Recall



Point Clé

Choisir la métrique selon le contexte :

- Haute Precision → éviter faux positifs (diagnostic médical)
- Haut Recall → attraper tous les cas (détection fraude)
- F1-Score → compromis équilibré

MERCI