TOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Scikit-learn

Sommaire

- Les métriques
- Optimisation hyperparamètres

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Les Métriques en Scikit-Learn

Introduction

Les métriques sont essentielles pour évaluer la performance des modèles de machine learning. Scikit-learn propose une large gamme de métriques adaptées aux différents types de problèmes : classification, régression, clustering, etc.

Métriques de Classification

Accuracy (Exactitude)

Définition : Proportion de prédictions correctes parmi toutes les prédictions.
Pour quels modèles : Tous les modèles de classification (Logistic Regression, SVM, Decision Trees, Random Forest, KNN, Naive Bayes)
Pourquoi : Métrique simple et intuitive qui donne une vision globale de la performance. Idéale quand toutes les classes ont la même importance et sont équilibrées.

```
accuracy = accuracy_score(y_true, y_pred)
```

Formule :

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

Avantages :

- Simple à comprendre et à interpréter
- Métrique intuitive

Limites :

- Peu adaptée aux datasets déséquilibrés
- Ne distingue pas les types d'erreurs

Matrice de Confusion

Définition : Tableau qui compare les prédictions aux vraies valeurs.

```
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(cm, display_labels=['Classe 0', 'Classe 1'])
disp.plot()
```

Structure (pour une classification binaire) :

```
                Prédiction
              |  Négatif  |  Positif  |
Réalité ------+-----------+-----------+
Négatif       |    TN     |    FP     |
Positif       |    FN     |    TP     |
```

Precision

Définition : Proportion de vrais positifs parmi toutes les prédictions positives.

Pour quels modèles : Logistic Regression, Random Forest, SVM, XGBoost (avec ajustement de seuil)

Pourquoi : Critique quand le coût d'un faux positif est élevé. On veut être sûr que ce qu'on prédit comme positif l'est vraiment. Permet d'optimiser le modèle pour minimiser les fausses alertes.

```
precision = precision_score(y_true, y_pred)
```

Formule :

```
Precision = TP / (TP + FP)
```

Precision

Interprétation : "Parmi tous les cas que j'ai prédits comme positifs, combien le sont réellement ?"

Quand l'utiliser : Lorsque le coût des faux positifs est élevé (ex: détection de spam, diagnostic médical).

Cas d'usage concrets :

- **Détection de spam** : Éviter de marquer un email important comme spam (faux positif coûteux)
- **Diagnostic médical** : Éviter les fausses alertes qui génèrent stress et examens inutiles
- **Système de modération** : Ne pas bloquer du contenu légitime

Recall

Définition : Proportion de vrais positifs parmi tous les cas réellement positifs.

Pour quels modèles : Random Forest, Gradient Boosting, Neural Networks (avec ajustement de seuil vers le bas)

Pourquoi : Essentiel quand manquer un cas positif est inacceptable (faux négatif coûteux). On privilégie capturer tous les positifs, même au prix de quelques fausses alertes. Permet d'optimiser la détection complète.

Recall

```
recall = recall_score(y_true, y_pred)
```

Formule :

```
Recall = TP / (TP + FN)
```

Interprétation : "Parmi tous les cas réellement positifs, combien ai-je réussi à identifier ?"

Quand l'utiliser : Lorsque le coût des faux négatifs est élevé (ex: détection de maladies graves, fraude bancaire).

Recall

Cas d'usage concrets :

- **Détection de fraude bancaire** : Identifier toutes les transactions frauduleuses
- **Détection d'intrusion informatique** : Capturer toutes les tentatives d'attaque
- **Recherche et sauvetage** : Ne manquer aucune personne en danger
- **Contrôle qualité critique** : Détecter tous les produits défectueux

F1-Score

Définition : Moyenne harmonique de la précision et du recall.

Pour quels modèles : XGBoost, LightGBM, Neural Networks sur données déséquilibrées, ensemble methods

Pourquoi : Trouve le meilleur compromis entre Precision et Recall. Particulièrement utile quand les classes sont déséquilibrées car il ne se laisse pas tromper par une accuracy artificielle. La moyenne harmonique pénalise les valeurs extrêmes (un modèle avec 100% Precision et 10% Recall aura un F1-Score faible).

F1-Score

```
f1 = f1_score(y_true, y_pred)
```

Formule :

```
F1 = 2 × (Precision × Recall) / (Precision + Recall)
```

Avantages :

- Équilibre entre précision et rappel
- Utile pour les datasets déséquilibrés

Quand l'utiliser : Lorsque vous voulez un équilibre entre précision et rappel, ou quand les classes sont déséquilibrées.

F1-Score

Cas d'usage concrets :

- **Classes déséquilibrées** : 1% de fraudeurs vs 99% de clients normaux
- **Diagnostic médical** : Balance entre détecter les malades et ne pas alarmer
- **Détection d'anomalies** : Peu d'anomalies dans beaucoup de données normales
- **Marketing ciblé** : Petit segment de clients à forte valeur

Specificity

Définition : Proportion de vrais négatifs parmi tous les cas réellement négatifs.

Pour quels modèles : Tous modèles de classification, surtout en médical et biologie

Pourquoi : Complémentaire au Recall. Mesure la capacité à identifier les vrais négatifs (personnes saines, transactions normales). Important pour éviter de surcharger le système avec trop de faux positifs. Utilisée en conjonction avec Recall pour équilibrer détection et spécificité.

Specificity

```
cm = confusion_matrix(y_true, y_pred)
tn, fp, fn, tp = cm.ravel()
specificity = tn / (tn + fp)
```

Formule :

```
Specificity = TN / (TN + FP)
```

Interprétation : Capacité du modèle à identifier correctement les cas négatifs.

Precision-Recall Curve

Définition : Courbe qui trace la précision contre le rappel pour différents seuils.

```
precision, recall, thresholds = precision_recall_curve(y_true, y_proba)
display = PrecisionRecallDisplay(precision=precision, recall=recall)
display.plot()
```

Quand l'utiliser : pour les datasets très déséquilibrés.

Classification Report

Définition : Rapport complet avec plusieurs métriques.

```
report = classification_report(y_true, y_pred, target_names=['Classe 0', 'Classe 1'])
print(report)
```

Contenu :

- Precision, Recall, F1-score par classe
- Support (nombre d'échantillons par classe)
- Moyennes

TOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Métriques de Régression

Mean Absolute Error (MAE)

Définition : Moyenne des valeurs absolues des erreurs.

Pour quels modèles : Linear Regression, Decision Trees, Random Forest, KNN pour régression

Pourquoi : Facile à interpréter car dans la même unité que la variable cible. Robuste aux outliers car utilise la valeur absolue (pas de carré). Donne une idée directe de l'erreur moyenne à attendre. Idéale pour communiquer avec des non-techniques.

```
mae = mean_absolute_error(y_true, y_pred)
```

Formule :

```
MAE = 1/n × Σ |y_i - ŷ_i|
```

Mean Absolute Error (MAE)

Avantages :

- Facile à interpréter (même unité que y)
- Robuste aux outliers

Cas d'usage concrets :

- **Prédiction de prix immobiliers** : "Le modèle se trompe en moyenne de 15 000€"
- **Prévision de température** : Erreur facilement compréhensible en degrés
- **Estimation de temps de trajet** : Erreur en minutes

Mean Squared Error (MSE)

Définition : Moyenne des carrés des erreurs.

Pourquoi : Pénalise beaucoup plus les grandes erreurs (au carré). Fonction différentiable partout, donc optimale pour l'entraînement des modèles (gradient descent). Utilisée comme fonction de perte dans la plupart des algorithmes d'optimisation. Préférable quand les grandes erreurs sont inacceptables.

```
mse = mean_squared_error(y_true, y_pred)
```

Formule :

```
MSE = 1/n × Σ (y_i - ŷ_i)²
```

Mean Squared Error (MSE)

Avantages :

- Pénalise davantage les grandes erreurs
- Dérivable (utile pour l'optimisation)

Limites :

- Unité au carré (difficile à interpréter)
- Sensible aux outliers

Cas d'usage concrets :

- **Optimisation de modèles** : Fonction dérivable utilisée dans l'apprentissage

Root Mean Squared Error (RMSE)

Définition : Racine carrée du MSE.

Pourquoi : Combine les avantages du MSE (pénalise grandes erreurs) avec l'interprétabilité du MAE (même unité que y). Très utilisée en pratique car elle est à la fois optimisable mathématiquement et compréhensible.

```
rmse = np.sqrt(mean_squared_error(y_true, y_pred))
# ou
rmse = mean_squared_error(y_true, y_pred, squared=False)
```

Formule :

```
RMSE = √(MSE)
```

Root Mean Squared Error (RMSE)

Avantages :

- Même unité que y (interprétable)
- Pénalise les grandes erreurs

Cas d'usage concrets :

- **Prévision de température** : Une erreur de 20°C est bien plus grave que 2 fois 10°C
- **Prédiction de demande** : Les grandes erreurs causent ruptures de stock ou surstock coûteux
- **Prévision de charge électrique** : Grandes erreurs peuvent causer des pannes

R^2 Score (Coefficient de Détermination)

Définition : Proportion de variance expliquée par le modèle.

Pour quels modèles : Tous modèles de régression (Linear, Ridge, Lasso, Random Forest, XGBoost)

Pourquoi : Métrique universelle pour comparer des modèles car sans unité (entre 0 et 1). Indique la qualité de l'ajustement : proche de 1 = excellent, proche de 0 = mauvais. Permet de répondre à "Est-ce que mes features sont prédictives ?". Très utilisée pour la sélection de modèles et features.

```
r2 = r2_score(y_true, y_pred)
```

R^2 Score (Coefficient de Détermination)

Formule :

$$R^2 = 1 - (SS_{res} / SS_{tot})$$

où $SS_{res} = \Sigma(y_i - \hat{y}_i)^2$ et $SS_{tot} = \Sigma(y_i - \bar{y})^2$

R^2 Score (Coefficient de Détermination)

Interprétation :

- **$R^2 = 1$** : Prédictions parfaites
- **$R^2 = 0$** : Modèle pas meilleur que la moyenne
- **$R^2 < 0$** : Modèle pire que la moyenne

Avantages :

- Sans unité (facile à comparer)
- Intuitivement interprétable

R^2 Score (Coefficient de Détermination)

Cas d'usage concrets :

- **Comparer Linear Regression (R^2=0.65) vs Random Forest (R^2=0.85)** : Le RF capture 85% de la variance
- **Évaluer la qualité prédictive** : R^2 proche de 1 = excellent, proche de 0 = modèle inutile
- **Analyse exploratoire** : Savoir si les features ont du pouvoir prédictif
- **Reporting business** : Métrique sans unité facile à communiquer

Mean Absolute Percentage Error (MAPE)

Définition : Erreur absolue moyenne en pourcentage.

Pourquoi : Exprime l'erreur en pourcentage, ce qui est universellement compréhensible. Permet de comparer la performance sur des datasets avec des échelles différentes. Idéale pour la communication avec le business (dire "8% d'erreur" est plus parlant que "500 unités"). Attention aux valeurs proches de zéro qui peuvent causer des problèmes.

```
mape = mean_absolute_percentage_error(y_true, y_pred)
```

Formule :

```
MAPE = 100/n × Σ |y_i - ŷ_i| / |y_i|
```

Mean Absolute Percentage Error (MAPE)

Avantages :

- Exprimé en pourcentage (facile à interpréter)

Limites :

- Problème si y contient des zéros
- Asymétrique (pénalise plus les sous-estimations)

Mean Absolute Percentage Error (MAPE)

Cas d'usage concrets :

- **Prévision de ventes** : "Le modèle se trompe de 8% en moyenne"
- **Prédiction de trafic web** : Erreurs relatives importantes et comparables
- **Forecasting financier** : Pourcentages compréhensibles
- **KPI business** : Communication facile avec les non-techniques

Note : Ne pas utiliser si les valeurs réelles peuvent être proches de zéro.

Métriques pour la Validation Croisée

Cross-Validation Score

```
# Pour la classification
scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')

# Pour la régression
scores = cross_val_score(model, X, y, cv=5, scoring='r2')

print(f"Scores: {scores}")
print(f"Moyenne: {scores.mean():.3f} (+/- {scores.std() * 2:.3f})")
```

Scoring Parameters

Classification :

- `'accuracy'` : Accuracy
- `'precision'` : Precision
- `'recall'` : Recall
- `'f1'` : F1-score

Régression

- `'r2'` : R^2 Score
- `'neg_mean_absolute_error'` : MAE (négatif)
- `'neg_mean_squared_error'` : MSE (négatif)
- `'neg_root_mean_squared_error'` : RMSE (négatif)

Note : Les métriques sont négatives car scikit-learn maximise toujours le score.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Choisir la Bonne Métrique

Pour la Classification

Contexte	Métrique Recommandée
Classes équilibrées	Accuracy
Classes déséquilibrées	F1-Score, ROC-AUC
Coût FP élevé	Precision
Coût FN élevé	Recall
Besoin d'équilibre	F1-Score
Prédictions probabilistes	Log Loss, ROC-AUC

Pour la Régression

Contexte	Métrique Recommandée
Interprétabilité	MAE, RMSE
Pénaliser grandes erreurs	MSE, RMSE
Comparer modèles	R^2
Erreur en %	MAPE
Robustesse aux outliers	MAE

Bonnes Pratiques

1. **Éviter le Biais d'Évaluation**

- **Toujours évaluer sur un ensemble de test séparé**
- **Utiliser la validation croisée**
- Ne jamais évaluer sur les données d'entraînement

Bonnes Pratiques

2. **Utiliser Plusieurs Métriques**

- Ne vous fiez pas à une seule métrique
- Utilisez `classification_report` pour une vue d'ensemble
- Visualisez la matrice de confusion

Bonnes Pratiques

3. **Considérer le Contexte Métier**

- Quel est le coût des différents types d'erreurs ?
- Quelles sont les contraintes du projet ?
- Quelle métrique a du sens pour le métier ?

Conclusion

Le choix de la métrique est crucial et dépend fortement du contexte du problème. Une bonne compréhension des métriques vous permettra de :

- Évaluer correctement vos modèles
- Comparer différentes approches
- Communiquer efficacement les performances
- Prendre des décisions éclairées

Conseil final : Commencez toujours par comprendre votre problème métier avant de choisir une métrique !

Merci pour votre attention

Des questions ?