



Modélisation et évaluation des modèles

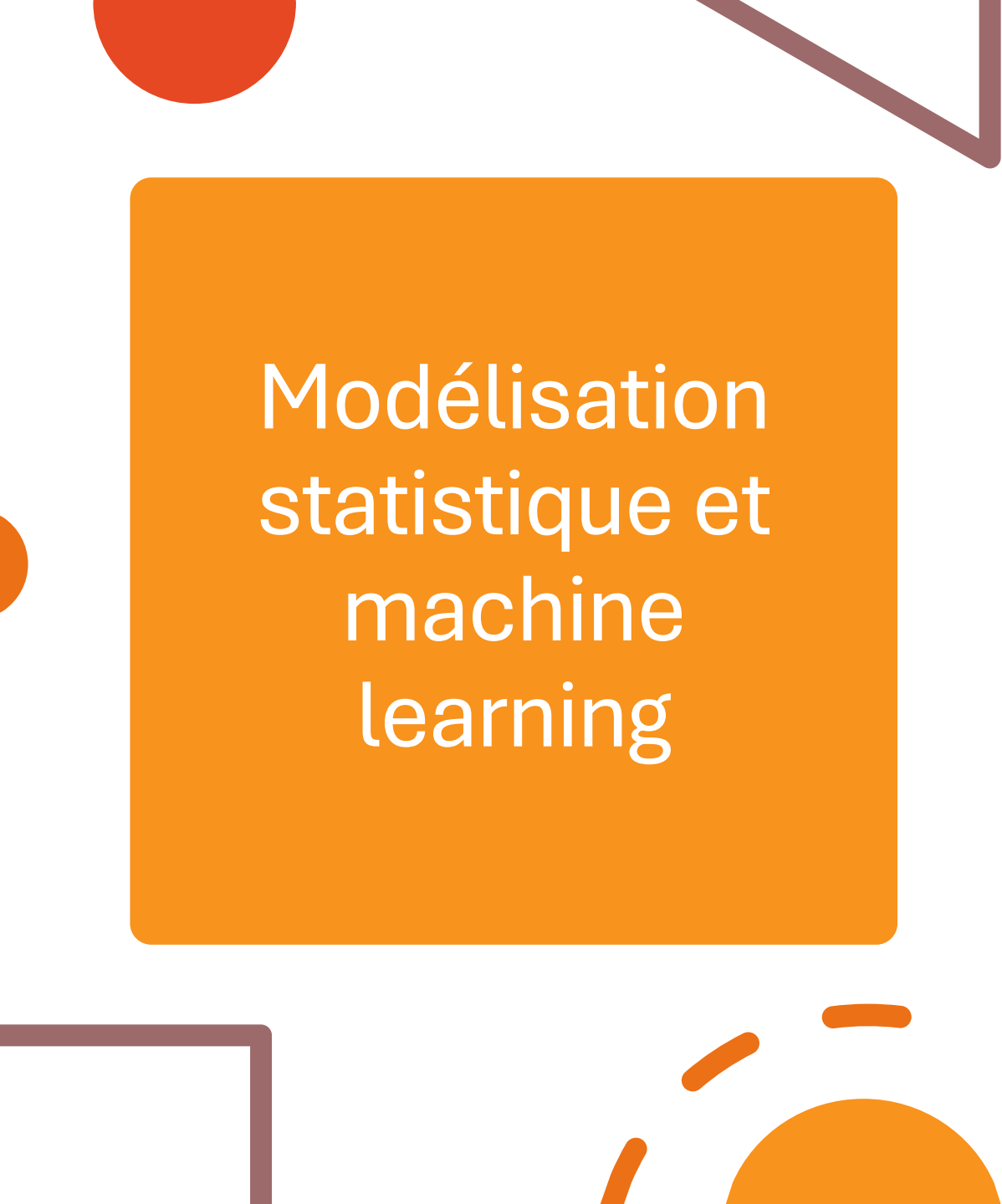


Amina MARIE

Objectifs

- Comprendre les bases de la modélisation statistique et de l'apprentissage automatique (ML).
- Construire des modèles prédictifs avec des algorithmes simples comme la régression linéaire.
- Évaluer la performance des modèles en utilisant des métriques appropriées.

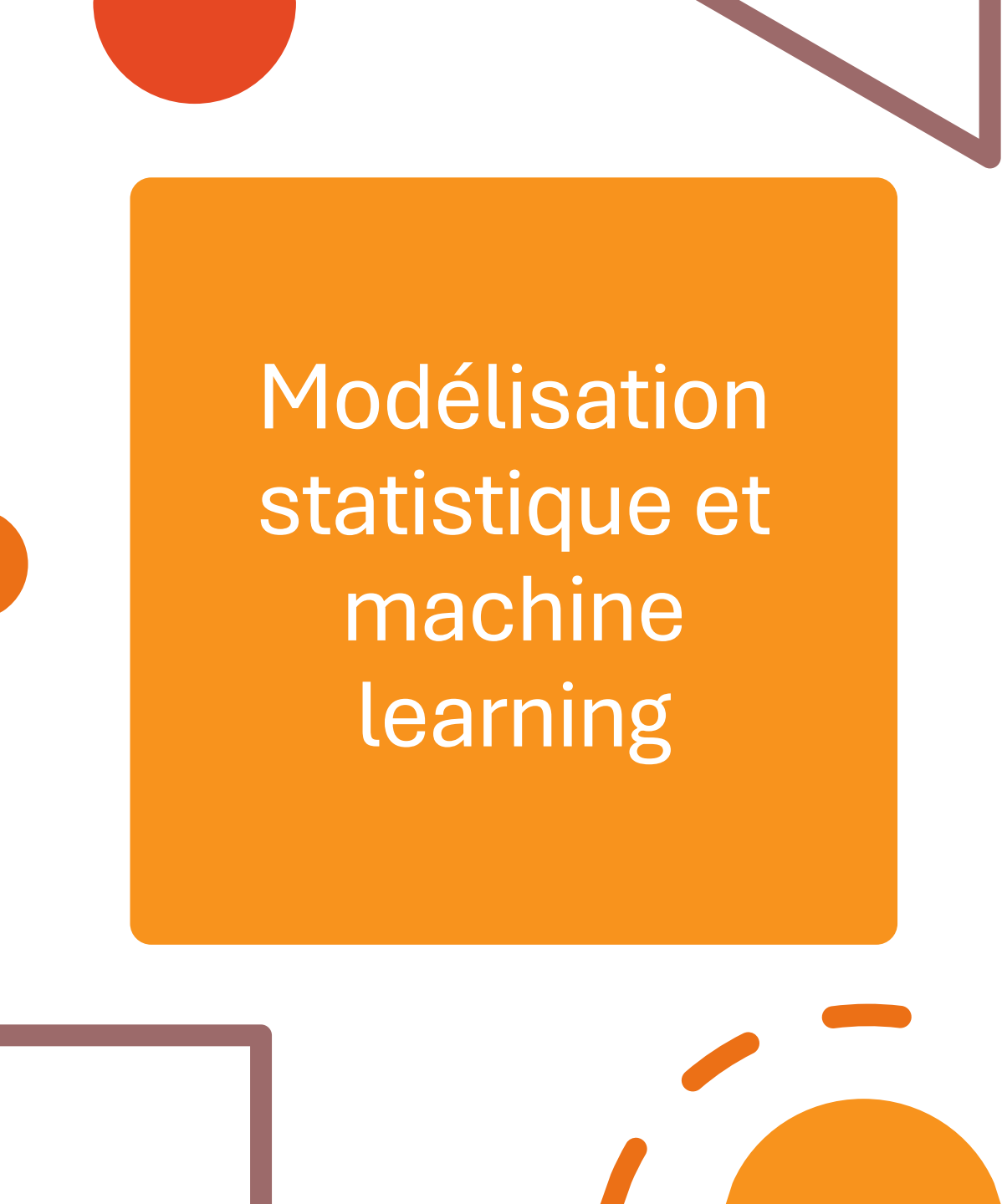
Introduction



Modélisation statistique et machine learning

Modélisation statistique : Utilisation de modèles mathématiques pour représenter les relations entre les variables.

Exemple : Régression linéaire pour prédire une température en fonction de l'humidité.



Modélisation statistique et machine learning

Apprentissage automatique (Machine

Learning) : Enseigner à une machine à prédire des résultats ou découvrir des motifs à partir des données.

- **Supervisé** : Le modèle apprend à partir de données étiquetées (prédiction de températures).
- **Non supervisé** : Le modèle découvre des motifs dans des données non étiquetées (clustering des régions climatiques).



Catégorie d'algorithmes

- **Régression** : Modèles pour des sorties continues.

Exemple : Prédire une température exacte.

- **Classification** : Modèles pour des sorties discrètes (catégories).

Exemple : Classer une journée comme "chaude" ou "froide".

- **Clustering** : Découverte de groupes similaires.

Exemple : Identifier des zones climatiques.





Pipeline typique d'un projet ML

1. Collecte et nettoyage des données.
2. Exploration des relations et tendances.
3. Construction d'un modèle (entraîner le modèle sur les données).
4. Évaluation des performances.
5. Déploiement et amélioration continue

Construction de modèles prédictifs

Regression linéaire

- **Objectif :** Construire un modèle simple pour prédire une variable continue (par exemple, la température).
- **Concept :** Trouver la droite qui minimise l'erreur entre les valeurs observées et les prédictions.
- **Utilisation :** Approprié pour des relations linéaires entre les variables.
- **Exemple Pratique : Prédire la Température**



Regression linéaire

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
3 from sklearn.metrics import mean_squared_error
4
5 # Données
6 X = data[['Precipitation', 'Humidity']] # Features
7 y = data['Temperature'] # Target
8
9 # Split des données
10 X_train, X_test, y_train, y_test = train_test_split(X, y,
11     test_size=0.2, random_state=42)
12
13 # Modèle
14 model = LinearRegression()
15 model.fit(X_train, y_train)
16
17 # Prédiction
18 y_pred = model.predict(X_test)
19
20 # Évaluation
21 rmse = mean_squared_error(y_test, y_pred, squared=False)
22 print(f"RMSE : {rmse}")
23
```

Regression linéaire

Chargement et préparation des données

- **Features (X)** : Ce sont les variables utilisées pour prédire la cible.

Ici, Precipitation et Humidity représentent les précipitations et l'humidité.

- **Target (y)** : C'est la variable que l'on souhaite prédire. Ici, c'est la Temperature.

La régression linéaire cherche à établir une relation entre les colonnes de X et la colonne y.

Regression linéaire

Pourquoi séparer les données ?

Pour évaluer la performance du modèle sur des données jamais vues pendant son entraînement.


- **Ensemble d'entraînement (X_{train} , y_{train})** : Sert à entraîner le modèle.
- **Ensemble de test (X_{test} , y_{test})** : Sert à mesurer les performances après l'entraînement.

Paramètres de `train_test_split` :

- `test_size=0.2` : 20 % des données sont réservées pour le test.
- `random_state=42` : Assure que la division reste identique à chaque exécution pour la reproductibilité.


Regression linéaire

- **Création du modèle :**
LinearRegression() instancie un modèle de régression linéaire prêt à être entraîné.
- **Entraînement (fit)**
- **La prédiction**
y_pred contient les températures prédites par le modèle.
- **Evaluation des performances avec le RMSE**



Arbre de décision

- **Objectif** : Effectuer des prédictions en segmentant les données en fonction de règles successives.
- **Concept** : Découper les données en sous-groupes basés sur des conditions logiques (ex : "Si humidité > 70, alors température > 20°C").
- **Utilisation** : Approprié pour des relations non linéaires ou des données complexes.
- **Exemple Pratique** : Prédire la Température



Arbre de décision

```
from sklearn.tree import DecisionTreeRegressor

# Modèle
tree_model = DecisionTreeRegressor(max_depth=4, random_state=42)
tree_model.fit(X_train, y_train)

# Prédiction
y_pred_tree = tree_model.predict(X_test)

# Évaluation
rmse_tree = mean_squared_error(y_test, y_pred_tree, squared=False)
print(f"RMSE (Arbre de décision) : {rmse_tree}")
```

Arbre de décision

- On charge et prépare les données comme précédemment avec features et target
- On sépare les données d'entraînement et de test
- Création du modele: `DecisionTreeRegressor`
 - `Max_depth = 4` est la limite de profondeur de l'arbre pour éviter un sur-apprentissage (overfitting)
- Prédiction: On applique les conditions qu'il a apprises à chaque ligne de `X_test` pour prédire les températures (`y_pred_tree`)
- Evaluation : on utilise RMSE pour mesurer la prédiction du modèle

Support Vector Machine (SVM)

- **Objectif** : Trouver une frontière optimale pour séparer les données ou effectuer des prédictions.
- **Concept** : Maximiser la distance entre la frontière et les points les plus proches (appelés vecteurs supports).
- **Utilisation** : Approprié pour des relations complexes ou des classes bien définies.
- **Exemple Pratique** : Classification des Jours



Support Vector Machine (SVM)

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Préparation des données pour la classification
data['Is_Hot'] = (data['Temperature'] > 25).astype(int)
# 1 pour chaud, 0 pour froid
X = data[['Precipitation', 'Humidity']]
y = data['Is_Hot']

# Split des données
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2, random_state=42)

# Modèle SVM
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

# Prédiction
y_pred_svm = svm_model.predict(X_test)

# Évaluation
accuracy = accuracy_score(y_test, y_pred_svm)
print(f"Précision du SVM : {accuracy}")
```



Construction de modèles prédictifs

Modèle	Application	Avantages	Inconvénients
Régression Linéaire	Régressions continues	Simple à comprendre et rapide	Performances limitées pour relations non linéaires.
Arbre de Décision	Régressions ou classifications	Interprétable, gère des relations complexes	Tendance au sur-apprentissage.
SVM	Classifications complexes	Bonne précision pour petites données	Plus lent avec de grandes bases de données.

Evaluation des performances

Evaluation des performances

Métriques pour la Régression

- **RMSE (Root Mean Squared Error)** : Mesure la différence moyenne entre les valeurs prédites et réelles.
 - Formule :
$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}$$
 - Interprétation : Plus c'est bas, mieux c'est.
- **R² (Coefficient de Détermination)** : Mesure la proportion de variance expliquée par le modèle.
 - Valeurs entre 0 et 1 (1 étant parfait).

Evaluation des performances

Métriques pour la Classification :

- **Précision**

La **précision** mesure la proportion des prédictions positives qui sont effectivement correctes. Elle répond à la question :

"Parmi toutes les prédictions positives faites par le modèle, combien sont réellement positives ?"

$$\text{Précision} = \frac{\text{Vrai Positifs (VP)}}{\text{Vrai Positifs (VP)} + \text{Faux Positifs (FP)}}$$

Evaluation des performances

Métriques pour la Classification :

- **Précision**

Exemple :

- Vous développez un modèle pour détecter des fraudes bancaires.
- **Vrai Positifs (VP)** : Transactions détectées comme frauduleuses et qui le sont vraiment.
- **Faux Positifs (FP)** : Transactions détectées comme frauduleuses mais qui sont légitimes.

Une faible précision signifie :

Le modèle fait beaucoup de fausses alertes (FP). Dans cet exemple, cela pourrait agacer les clients en bloquant à tort des transactions légitimes.

Evaluation des performances

Métriques pour la Classification :

- **Rappel**

Le **rappel** mesure la proportion des cas positifs correctement identifiés par le modèle. Il répond à la question :

"Parmi toutes les vraies occurrences positives, combien le modèle en a-t-il détecté ?"

$$\text{Rappel} = \frac{\text{Vrai Positifs (VP)}}{\text{Vrai Positifs (VP)} + \text{Faux Négatifs (FN)}}$$

Evaluation des performances

Métriques pour la Classification :

- **Rappel**

Exemple :

Toujours dans le cas de la fraude bancaire :

- **Vrai Positifs (VP)** : Transactions frauduleuses correctement détectées.
- **Faux Négatifs (FN)** : Transactions frauduleuses non détectées.
- **Un faible rappel signifie :**
Le modèle manque de nombreuses fraudes (FN). Cela pourrait entraîner des pertes financières importantes pour l'entreprise.

Evaluation des performances

Métriques pour la Classification :

- **F-Mesure**

La **F-mesure** est la moyenne harmonique de la Précision et du Rappel. Elle combine ces deux métriques pour donner une vue globale de la performance du modèle.

$$\text{F-mesure} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Interprétation :

- La F-mesure est utile lorsque vous avez besoin d'un équilibre entre Précision et Rappel.
- Si l'une des deux métriques est beaucoup plus faible que l'autre, la F-mesure en sera fortement affectée.

Evaluation des performances

Exemple d'interprétation :

Imaginons un modèle de détection de spams qui donne les résultats suivants :

- **Précision = 0.85** (85 % des e-mails marqués comme spam sont vraiment des spams).
- **Rappel = 0.70** (70 % de tous les spams présents dans la boîte de réception sont correctement détectés).

Cas favorable : Si vous voulez éviter les faux positifs (éviter de marquer des e-mails légitimes comme spam), concentrez-vous sur la **Précision**.

Cas critique : Si vous voulez détecter autant de spams que possible, concentrez-vous sur le **Rappel**, quitte à avoir quelques fausses alertes.

Équilibre : La F-mesure peut aider à trouver un compromis dans une situation où les deux métriques sont importantes.

Evaluation des performances

Métriques pour la Classification :

Métrique	Question clé	Cas d'utilisation prioritaire
Précision	Parmi les positifs prédits, combien sont corrects ?	Lorsque les faux positifs doivent être minimisés.
Rappel	Parmi les vrais positifs, combien sont détectés ?	Lorsque les faux négatifs doivent être minimisés.
F-mesure	Quel est l'équilibre entre Précision et Rappel ?	Lorsque vous cherchez un compromis entre Précision et Rappel.



Evaluation des performances

```
from sklearn.metrics import precision_score, recall_score, f1_score

# Précision
precision = precision_score(y_test, y_pred_svm)
recall = recall_score(y_test, y_pred_svm)
f1 = f1_score(y_test, y_pred_svm)

print(f"Précision : {precision}")
print(f"Rappel : {recall}")
print(f"F-mesure : {f1}")
```

Exercices

Exercices

Livrables attendus

1. Modèle de régression linéaire entraîné et évalué.
2. Analyse des clusters et visualisation des groupes.
3. Rapport d'interprétation des résultats (RMSE, R^2 , visualisation des clusters).



Exercice 1

Modélisation de la température avec Régression Linéaire

Votre entreprise souhaite prédire les températures en fonction d'autres facteurs climatiques (comme les précipitations, l'humidité, etc.). Vous êtes chargé de construire un modèle de régression linéaire pour cette tâche.

Etape 1

Chargement des données

Consigne : Chargez les données et préparez-les pour la modélisation.

- Importez le fichier `weather_data_transformed.csv`.
- Séparez les colonnes en **features (X)** et **target (y)**, où la cible est la colonne Temperature.

Jeu d'entraînement 80%, jeu de test 20%



Etape 2

Normalisation des données

Consigne : Normalisez les données avec **StandardScaler** pour les rendre comparables.



Etape 3

Création et entraînement du modèle

Objectif : Construire un modèle de régression linéaire et l'entraîner sur les données normalisées.

Etape 4

Évaluation des performances

Consigne : Utilisez les métriques RMSE et R^2 pour évaluer les performances du modèle.


Quelle est la performance du modèle en termes de RMSE ?



Exercice 2

K-Means Clustering pour la Classification des Régions Climatiques

Votre équipe veut identifier des régions similaires sur la base des données climatiques. Vous utilisez l'algorithme **K-Means** pour regrouper ces régions en clusters.





Etape 1

Préparation des données


Consigne : Sélectionnez les colonnes pertinentes pour le clustering (Temperature, Precipitation, etc.).



Etape 2

Construction du modèle K-Means

Consigne : Appliquez K-Means avec 3 clusters et affichez les groupes résultants.





Etape 3

Visualisation des clusters

Consigne : Utilisez un nuage de points pour visualiser les clusters obtenus.

