

# Plateforme Web de Machine Learning

## Membres du groupe :

- Abderrahim Aamirrou & Imane Chalati & Halima Driouch

## Encadrant du projet :

- Professeur Mohammed AMEKSA

## Filière , Semestre et Année universitaire :

- IASD - 5<sup>e</sup> semestre - 2025 – 2026

## **1. Sujet du projet :**

- Le projet consiste à concevoir une plateforme web interactive qui permet aux utilisateurs de charger différents types de datasets, comme ceux utilisés dans nos cas pratiques ([\*fitness\\_dataset.csv\*](#), [\*calories\\_pred.csv\*](#)), et de réaliser des analyses, entraîner des modèles et effectuer des prédictions intelligentes sans avoir besoin de coder.
- **Dataset de Régression** — Calcul des Calories : Ce dataset contient des données numériques qui permettent de **prédire le nombre total de calories consommées** par une personne.
- **Dataset de Classification** — Fit ou Non Fit : Ce dataset contient des informations sur les calories consommées + d'autres paramètres et attribue à chaque personne une étiquette : **Fit ou Non Fit**.

## **2. Description de la Plateforme FitAI :**

**FitAI** est une plateforme interactive développée par notre groupe dans le cadre de l'étude de l'apprentissage supervisé.

Elle a été conçue pour rendre l'entraînement, l'analyse et la comparaison des modèles de Machine Learning accessibles, simples et intuitifs.

L'objectif principal de **FitAI** est de permettre aux utilisateurs d'explorer les deux branches majeures de l'apprentissage supervisé :

- la régression,
- la classification.

Grâce à une interface claire et organisée, **FitAI** offre une expérience complète allant du traitement des données à l'interprétation des résultats.

**FitAI** permet à l'utilisateur d'insérer ses propres données, de choisir l'un des modèles disponibles, puis la plateforme calcule automatiquement les calories et prédit si la personne est Fit ou Non Fit selon les informations fournies.

### **3. Outils et Technologies Utilisés dans FitAI**

Selon les recommandations de notre professeur, pour le développement de la plateforme **FitAI**, notre groupe a sélectionné des outils et technologies modernes afin de créer une application **fiable, interactive et pédagogique**. Ces choix ont permis de combiner **la puissance des modèles de Machine Learning avec une expérience utilisateur claire et intuitive**, facilitant à la fois l'expérimentation et l'apprentissage.

- **Algorithmes de Machine Learning** : Regression Linéaire, SVR, Arbre de Décision, Random Forest, XGBoost, Regression Logistique et SVM, pour effectuer des prédictions précises et classer les utilisateurs en Fit / Non Fit.
- **Fichiers .pkl** : Modèles entraînés enregistrés pour être réutilisés rapidement sans réentraînement.
- **Django** : Framework Python utilisé pour le back-end, la gestion des modèles et l'authentification des utilisateurs.
- **Front-End (HTML, CSS , Django Templates)** : Interface interactive et intuitive permettant à l'utilisateur d'insérer ses données, choisir un modèle et visualiser les résultats.
- **Datasets CSV** : Données d'entraînement et de test utilisées par les modèles, téléchargeables pour analyse complémentaire.

### **4. Architecture de la Plateforme :**

#### **a. Structure hiérarchique**

**Le dossier principal** : Conteneur principal organisant les trois grands domaines du projet.

- **atelier** : Code de production
- **datasets** : Gestion des données
- **modeles avec code source ipynb** : Recherche et développement

## b. Application Django : « atelier »

Le dossier **atelier** constitue la partie principale du projet.

Il contient **l'environnement complet de développement**, ainsi que tous les fichiers nécessaires au fonctionnement de la plateforme de Machine Learning.

Il contient :

- Le fichier **requirements.txt** : Ce fichier regroupe toutes les bibliothèques Python utilisées dans le projet (Django, NumPy, scikit-learn, XGBoost...). Il permet d'installer rapidement les dépendances nécessaires sur un autre ordinateur.
- Le dossier **mlPlatform** : contient :
  - Les modèles pré-entraînés de Machine Learning (fichiers .pkl)
  - Les scripts Python qui gèrent la logique du site
  - Les fichiers HTML et CSS pour l'interface
  - La base de données SQLite (**db.sqlite3**)
  - Les templates, images, pdfs du support
  - Le fichier principal **manage.py** pour exécuter le serveur.
- Le dossier **migrations** : est un dossier généré automatiquement par Django. Il joue un rôle essentiel dans la gestion de la base de données. Et pour des fichiers Python créés automatiquement par Django chaque fois que vous modifiez votre modèle (models.py). Ils servent à **traduire les changements du code vers la base de données SQLite**.
  - **0001\_initial.py** : première création des tables
  - **0002\_auto\_XXXX.py** : modifications automatiques après changement dans les modèles
  - **\_\_init\_\_.py** : permet à Django de reconnaître le dossier comme un module Python
- Le dossier **algoML** : regroupe toute la logique liée aux algorithmes de Machine Learning, ainsi que la structure Django qui permet de les utiliser dans le site web. Et il est responsable sur **Les fichiers principaux du module Django** qui gèrent le comportement de la plateforme (Backend Django) :
  - **views.py** : Contient le code qui reçoit les données de l'utilisateur, exécute les modèles ML et renvoie les résultats vers l'interface.
  - **urls.py** : Définit toutes les routes de navigation vers les pages de classification, régression, détails des modèles...
  - **models.py** : Gère la structure de la base de données (par exemple l'historique des prédictions).

- **admin.py** : Permet de gérer les données depuis l'interface d'administration de Django.
- **apps.py** : Fichier standard du fonctionnement d'une application Django.

### c. Machine Learning « ML Models »

- **Les modèles sont chargés depuis la plateforme via les fichiers .pkl.**

Classification	Régression
SVC	SVR
RandomForestClassifier	RandomForestRegressor
LogisticRegression	LinearRegression
XGBClassifier	XGBRegressor
DecisionTreeClassifier	DecisionTreeRegressor

## 5. GUIDE POUR DE NOTRE PLATEFORME PAR DES SCREENSHOTS SUIVIS PAR DES DESCRIPTIONS:

### a. PAGE D'ACCUEIL :

The screenshot shows the homepage of the FIT AI Platform. At the top, there is a navigation bar with the logo "FIT AI Platform", links for "Accueil" and "À propos", and buttons for "Connexion" and "Inscription". Below the navigation bar, the title "Dashboard IA & Fitness" is displayed in large bold letters, followed by the subtitle "Explorez nos algorithmes de Machine Learning pour la santé et la performance". A descriptive text states: "Cette plateforme vous permet de concevoir, d'entraîner et de déployer des modèles d'apprentissage automatique en toute simplicité.". The page is divided into two main sections: "Modèles de Régression" (yellow background) and "Modèles de Classification" (grey background). Each section contains a brief description and an example model. In the "Modèles de Régression" section, it says: "Explorez nos modèles de régression pour réaliser des prédictions continues (fréquence cardiaque, calories, durée d'effort)." and "Ex: régression linéaire, SVR". In the "Modèles de Classification" section, it says: "Entraînez vos modèles avec des algorithmes performants pour classer activités, anomalies ou états de santé." and "Ex: Random Forest, SVM". At the bottom of the page, there is a footer with the text "© 2025-2026 AI Platform. - All rights reserved."

"Interface d'accueil intuitive présentant les deux types principaux de problèmes ML disponibles : Classification et Régression. Navigation simple vers les différentes fonctionnalités."

### b. PAGE DE CONNEXION :

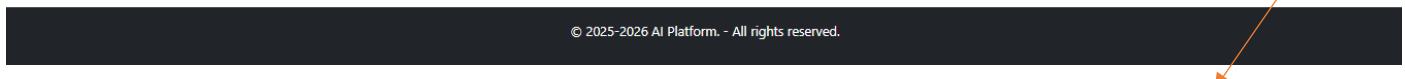


### Connexion

Nom d'utilisateur  
Mot de passe

**Accéder à mon espace**

Pas encore de compte ? [Créer un compte](#)



### Créer un compte FitAI

Nom d'utilisateur  
Adresse Email  
Mot de passe      Confirmation

**S'inscrire maintenant**

Déjà membre ? [Se connecter](#)



"Interface d'inscription simplifiée pour créer un compte utilisateur avec validation en temps réel."

c. INTERFACE ALGORITHMIQUE :

# Dashboard IA & Fitness

Explorez nos algorithmes de Machine Learning pour la santé

**Régression Linéaire**



IA & Machine Learning (M354)

Cours: Apprentissage Supervisé - Régression linéaire

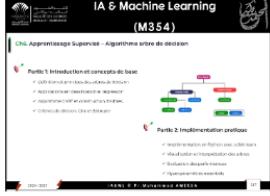
- Régression sur l'apprentissage supervisé
- Les notions fondamentales de l'apprentissage supervisé
- La régression linéaire

L'approche classique pour estimer une valeur continue (Calories) par une relation linéaire.

**Régression**

[Détails](#) [Tester ce modèle](#)

**Arbre de Décision (Régression)**



IA & Machine Learning (M354)

Cours: Apprentissage Supervisé - Algorithme arbre de décision

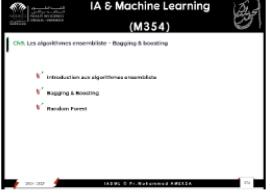
- Partie 1: Introduction et concepts de base
- Algorithme d'apprentissage supervisé
- Classification et régression
- Modélisation avec un arbre de décision
- Prise de décision et prévision

Modèle simple et interprétable qui suit des règles logiques pour déduire votre état.

**Régression**

[Détails](#) [Tester ce modèle](#)

**Random Forest (Régression)**



IA & Machine Learning (M354)

Cours: Les algorithmes ensemble - Bagging & Boosting

- Introduction aux algorithmes ensemble
- Bagging & Boosting
- Random Forest

Prédiction précise du nombre de Calories brûlées basée sur l'intensité de l'effort.

**Régression**

[Détails](#) [Tester ce modèle](#)

**SVR (Régression)**



IA & Machine Learning (M354)

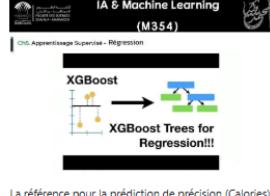
Cours: Apprentissage Supervisé - Régression

Utilise les vecteurs de support pour prédire une valeur continue avec une marge de tolérance (Tube).

**Régression**

[Détails](#) [Tester ce modèle](#)

**XGBoost (Régression)**



IA & Machine Learning (M354)

Cours: Apprentissage Supervisé - Régression

XGBoost

XGBoost Trees for Regression!!!

La référence pour la prédiction de précision (Calories). Rapide, performant et gagnant de compétitions.

**Régression**

[Détails](#) [Tester ce modèle](#)

© 2025-2026 AI Platform. - All rights reserved.

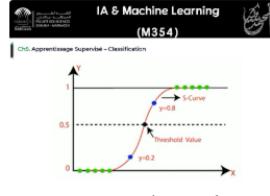
"L'interface affiche une galerie de 5 cartes algorithmiques disposées . Chaque carte représente un modèle de régression disponible. "

Et de même pour la classification :

# Dashboard IA & Fitness

Explorez nos algorithmes de Machine Learning pour la santé

**Régression Logistique**



IA & Machine Learning (M354)

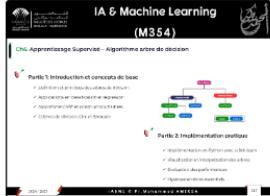
Cours: Apprentissage Supervisé - Classification

Classification binaire pour prédire si vous êtes FIT ou NOT FIT. Idéal pour les probabilités simples.

**Classification**

[Détails](#) [Tester ce modèle](#)

**Arbre de Décision(Classification)**



IA & Machine Learning (M354)

Cours: Apprentissage Supervisé - Algorithme arbre de décision

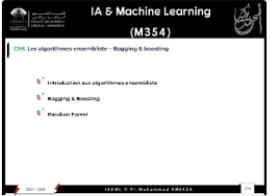
- Partie 1: Introduction et concepts de base
- Algorithme d'apprentissage supervisé
- Classification et régression
- Modélisation avec un arbre de décision
- Prise de décision et prévision

Modèle simple et interprétable qui suit des règles logiques pour déduire votre état.

**Classification**

[Détails](#) [Tester ce modèle](#)

**Random Forest (Classif)**



IA & Machine Learning (M354)

Cours: Les algorithmes ensemble - Bagging & Boosting

- Introduction aux algorithmes ensemble
- Bagging & Boosting
- Random Forest

Un ensemble d'arbres de décision pour une précision maximale sur votre état de forme.

**Classification**

[Détails](#) [Tester ce modèle](#)

**SVM (Support Vector)**

IA & Machine Learning (M354)

Trouve l'hyperplan optimal pour séparer les profils sportifs avec une grande rigueur mathématique.

**Classification**

Détails

Tester ce modèle

**XGBoost (Classif)**

IA & Machine Learning (M354)

L'algorithme de Boosting le plus puissant. Corrige séquentiellement les erreurs des arbres précédents.

**Classification**

Détails

Tester ce modèle

© 2025-2026 AI Platform. - All rights reserved.

## d. PAGE "À PROPOS" :

Fit AI Platform
Accueil À propos
Connexion Inscription

# À Propos de FitAI

L'alliance de la science des données et de la performance humaine

### Notre Mission

FitAI est une plateforme innovante née d'un projet académique ambitieux. Notre objectif est de démocratiser l'accès aux prédictions de santé précises grâce au Machine Learning.

Nous utilisons des algorithmes avancés (XGBoost, Random Forest, SVM) pour analyser des milliers de données et fournir des estimations fiables sur :

- La dépense calorique exacte
- L'état de forme physique (Fit/Not Fit)
- Les facteurs de risques santé



### Le Projet

**Cadre Académique**  
Développé dans le cadre du module "IA & Machine Learning".

**Technologie**  
Utilise Python, Django, Scikit-Learn et XGBoost.

**Innovation**  
Une approche "Data-Driven" pour le sport et la santé.

### Notre Équipe de Développement

Abderrahim Aamirrou

Imane Chalati

Halima Driouch

© 2025-2026 AI Platform. - All rights reserved.

**"Page informative présentant la plateforme ML, l'équipe de développement, et les technologies utilisées, avec design professionnel et sections organisées."**

## e. Détails des algorithmes :

The screenshot shows a web interface for the FIT AI Platform. At the top, there is a navigation bar with links for 'Accueil' and 'À propos', and buttons for 'Connexion' and 'Inscription'. Below the navigation bar, the page title is 'Régression Linéaire'. Underneath the title, there is a breadcrumb navigation 'Régression - Statistique'. The main content area is divided into two columns. The left column contains a section titled 'Qu'est-ce que c'est ?' which describes linear regression as a classic mathematical approach to find a straight line that best fits data points. It includes the formula  $\text{Calories} = a * \text{Durée} + b * \text{Rythme} + c$ . The right column contains a section titled 'Caractéristiques' listing pros and cons: pros include 'Simplicité' (ideal for simple trends), 'Coefficients' (knows exactly the impact of each variable), and 'MSE' (132 Calories); cons include 'Limite' (fails if the relationship is not linear) and 'MAE' (8.44 Calories). Below these sections, there are two separate document preview windows. The top window is titled 'Documentation Théorique' and shows a slide from a presentation on 'IA & Machine Learning' at SEMIAIA-MARRAKECH, featuring a logo of a stylized building. The bottom window also titled 'Documentation Théorique' shows a slide titled 'La régression Linéaire' from a course by Pr. Mohammed AMEKSIA for the 2025/2026 academic year. This slide includes a definition of supervised learning as a fundamental approach in ML that allows a machine to learn from examples labeled for prediction. The footer of the page includes a copyright notice: '© 2025-2026 AI Platform. - All rights reserved.'

" Pages spécialisées pour chaque algorithme avec explications théoriques et des pdfs de cours et pour les chapitres restant nous avons choisis des cours sur le web . "

## f. TESTEUR DE MODÈLES :

Pour Régression (prédiction des calories ):

Avant la prédiction :

## Atelier Pratique : Arbre de Décision (Rég)

Saisissez les paramètres ci-dessous pour lancer une prédiction en temps réel.

### Paramètres de simulation

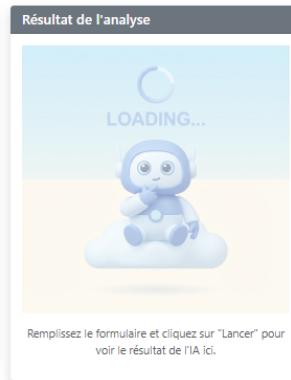
Données Physiques

Genre	Age (ans)	
Homme	Ex: 25	
Taille (cm)	Poids (kg)	Rythme Cardiaque (BPM)
Ex: 175	Ex: 70	Ex: 80

Détails de l'Exercice

Durée (minutes)	Température Corporelle (°C)
Ex: 45	Ex: 39.5

**LANCER LA PRÉDICTION**



© 2025-2026 AI Platform. - All rights reserved.

### Après le test :

## Atelier Pratique : Arbre de Décision (Rég)

Saisissez les paramètres ci-dessous pour lancer une prédiction en temps réel.

### Paramètres de simulation

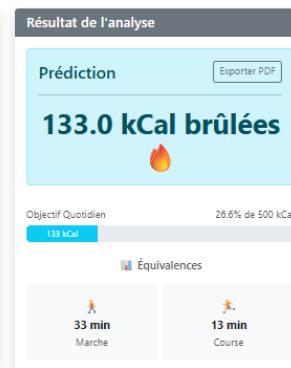
Données Physiques

Genre	Age (ans)	
Homme	Ex: 25	
Taille (cm)	Poids (kg)	Rythme Cardiaque (BPM)
Ex: 175	Ex: 70	Ex: 80

Détails de l'Exercice

Durée (minutes)	Température Corporelle (°C)
Ex: 45	Ex: 39.5

**LANCER LA PRÉDICTION**



© 2025-2026 AI Platform. - All rights reserved.

**" Affichage du résultat de régression : prédiction de calories brûlées avec équivalences en activités physiques et progression vers l'objectif quotidien. Ici l'exemple d'arbre de décision . "**

### Pour Classification (prédire si Fit ou Non Fit) :

### Avant le test :

## Atelier Pratique : Random Forest Classifier

Saisissez les paramètres ci-dessous pour lancer une prédiction en temps réel.

**Paramètres de simulation**

Données Physiques

Genre	Age (ans)	
Homme	Ex: 25	
Taille (cm)	Poids (kg)	Rythme Cardiaque (BPM)
Ex: 175	Ex: 70	Ex: 80

Habitudes de Vie

Heures de Sommeil	Qualité Nutrition (1-10)
7	5
Niveau d'Activité (1-5)	Fumeur ?
3	Non

**LANCER LA PRÉDICTION**

**Résultat de l'analyse**



Remplissez le formulaire et cliquez sur "Lancer" pour voir le résultat de l'IA ici.

### Après l'insertion du données :

Atelier Pratique : Random Forest Classifier

Saisissez les paramètres ci-dessous pour lancer une prédiction en temps réel.

**Paramètres de simulation**

Données Physiques

Genre	Age (ans)	
Homme	Ex: 25	
Taille (cm)	Poids (kg)	Rythme Cardiaque (BPM)
Ex: 175	Ex: 70	Ex: 80

Habitudes de Vie

Heures de Sommeil	Qualité Nutrition (1-10)
7	5
Niveau d'Activité (1-5)	Fumeur ?
3	Non

**LANCER LA PRÉDICTION**

**Résultat de l'analyse**

**Prédition**

**FIT** 💪

Score de Certitude du Modèle  
**73.14%**  
Certitude Moyenne

Niveau de certitude  
73.14%

Probabilités par Classe

FIT 73.14%	NOT FIT 26.86%
---------------	-------------------

" Affichage du résultat de classification : prédition "FIT" avec score de certitude de 73,14% et distribution des probabilités entre les deux classes possibles. Per exemple pour random forest . "

## 6. Les modèles de Machine Learning :

### a. Traitement des datasets :

## Pour la dataset de classification :

```
[5]: df = pd.read_csv('fitness_dataset.csv')
df.isnull().sum()

[5]: age          0
height_cm      0
weight_kg      0
heart_rate     0
blood_pressure 0
sleep_hours    160
nutrition_quality 0
activity_index 0
smokes         0
gender          0
is_fit          0
dtype: int64

[6]: df['sleep_hours'] = df['sleep_hours'].fillna(df['sleep_hours'].median())

[7]: df.isnull().sum()

[7]: age          0
height_cm      0
weight_kg      0
heart_rate     0
blood_pressure 0
sleep_hours    0
nutrition_quality 0
activity_index 0
smokes         0
gender          0
is_fit          0
dtype: int64

# Nettoyage de la colonne 'smokes'
# On force tout en chaîne de caractères pour uniformiser puis on remplace
df['smokes'] = df['smokes'].astype(str)

# Dictionnaire de remplacement
# car lorsqu'on fait directement l'encodage avec LabelEncoder il donne 4 valeurs 0/1/2/3
map_smokes = {
    'yes': 1,
    'no': 0,
    '1': 1,
    '0': 0
}

# On applique le mapping
df['smokes'] = df['smokes'].map(map_smokes)

# Encodage de 'gender'
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])
```

## Pour la dataset de Régression :

```
[2]: df = pd.read_csv('calories_pred.csv')

[3]: X = df.drop(columns=['Calories', 'Unnamed: 0', 'User_ID'])
y = df['Calories']

le = LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])

scaler = StandardScaler()

num_cols = ['Age', 'Height', 'Weight', 'Duration', 'Heart_Rate', 'Body_Temp']

X[num_cols] = scaler.fit_transform(X[num_cols])
```

## b. Les Modèles d' Apprentissage Supervisé

# REGRESSION LINIAIRE

```
[6]: lr = LinearRegression()
lr.fit(X_train, y_train)

prd = lr.predict(X_test)

[7]: mae = mean_absolute_error(y_test, prd)
mse = mean_squared_error(y_test, prd)
r2 = r2_score(y_test, prd)
print(mae)
print(mse)
print(r2)

8.44151355384971
131.99574575081698
0.9672937151257295

[8]: import joblib
joblib.dump(lr, 'linear_reg.pkl')

[8]: ['linear_reg.pkl']
```

## REGRESSION LOGISTIQUE

```
[14]: logi_reg = LogisticRegression()
logi_reg.fit(X_train, y_train)

pred = logi_reg.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, pred)*100:.4f}%")
print(f"precision: {precision_score(y_test, pred)*100:.4f}%")
print(f"f1_score: {f1_score(y_test, pred)*100:.4f}%")
print(f"recall: {recall_score(y_test, pred)*100:.4f}%")
```

Accuracy: 76.2500%  
precision: 74.3056%  
f1\_score: 69.2557%  
recall: 64.8485%

```
[17]: package = { "model": logi_reg, "scaler": scaler}
import joblib
joblib.dump(package, 'logistic_reg.pkl')
```

```
[17]: ['logistic_reg.pkl']
```

## DECISION TREE CLASSIFICATION

```
[18]: #model
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth=5, min_samples_leaf=4, min_samples_split=2,random_state=42)
clf.fit(X_train, y_train)

#evaluation
pred = clf.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, pred)*100:.4f}%")
print(f"precision: {precision_score(y_test, pred)*100:.4f}%")
print(f"f1_score: {f1_score(y_test, pred)*100:.4f}%")
print(f"recall: {recall_score(y_test, pred)*100:.4f}%")
```

Accuracy: 73.2500%  
precision: 71.6418%  
f1\_score: 64.2140%  
recall: 58.1818%

```
[11]: import joblib
joblib.dump(clf, 'decision_tree_classification.pkl')
```

```
[11]: ['decision_tree_classification.pkl']
```

## DECISION TREE REGRESSION 1

```
[7]: dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
pred = dt.predict(X_test)
```

```
[8]: mae = mean_absolute_error(y_test, pred)
mse = mean_squared_error(y_test, pred)
r2 = r2_score(y_test, pred)
```

```
[9]: print(mae)
print(mse)
print(r2)
```

3.3966666666666665  
27.842666666666666  
0.9931010641102141

```
[10]: import joblib
joblib.dump(dt, 'decision_tree_regression.pkl')
```

```
[10]: ['decision_tree_regression.pkl']
```

## RANDOM FOREST CLASSIFICATION

```
[12]: model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
model.fit(X, y)
```

```
[12]: ▾ RandomForestClassifier ⏎ ⏎
      ► Parameters
```

```
[13]: y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {acc*100:.4f}%")
print(f"precision: {precision_score(y_test, y_pred)*100:.4f}%")
print(f"f1_score: {f1_score(y_test, y_pred)*100:.4f}%")
print(f"recall: {recall_score(y_test, y_pred)*100:.4f}%")
```

Accuracy: 98.9130%  
precision: 98.6395%  
f1\_score: 98.6395%  
recall: 98.6395%

```
[14]: package = {
    'modele': model,
    'encoders': {'gender': le_gender, 'smokes': le_smokes},
    'colonnes_features': X.columns.tolist()
}

joblib.dump(package, 'rf_classification.pkl')
print("'rf_classification.pkl'")

'rf_classification.pkl'
```

## Random Forest REGRESSION

```
[7]: model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

```
[8]: mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
mse = mean_squared_error(y_test, predictions)

print(f"R2 Score : {r2:.4f}")
print(f"MAE : {mae:.2f}")
print(f"MSE : {mse:.2f}")
```

R2 Score : 0.9982  
MAE : 1.72  
MSE : 7.200539

```
[9]: import joblib
joblib.dump(model, 'rf_regression.pkl')
```

```
[9]: ['rf_regression.pkl']
```

## SVM CLASSIFICATION

```
[9]: model = SVC( kernel='rbf',C=1.0, gamma='scale',probability=True,random_state=42)
model.fit(X_train_scaled, y_train)
print("Modele SVM Classification entraîné !")

Modele SVM Classification entraîné !

[10]: y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {accuracy * 100:.2f}%")
print(f"precision: {precision_score(y_test, y_pred)* 100:.4f}%")
print(f"f1_score: {f1_score(y_test, y_pred)* 100:.4f}%")
print(f"recall: {recall_score(y_test, y_pred)* 100:.4f}%")

Accuracy: 82.07%
precision: 80.4511%
f1_score: 76.4286%
recall: 72.7891%

[11]: model_package = {'model': model,'scaler': scaler,'label_gender': le_gender,'label_smokes': le_smokes,'feature_columns': features}
joblib.dump(model_package, 'svm_class_model.pkl')
print("Sauvegarde terminée : svm_class_model.pkl")

Sauvegarde terminée : svm_class_model.pkl
```

## SVM REGRESSION

```
[7]: model = SVR(kernel='rbf',C=1.0,gamma='scale',epsilon=0.1)
model.fit(X_train_scaled, y_train)

[7]: ▾ SVR ⓘ ?
      ► Parameters

[8]: y_pred = model.predict(X_test_scaled)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"R² Score : {r2:.4f}")
print(f"RMSE : {rmse:.2f}")
print(f"MAE : {mae:.2f} ")
print(f"MSE : {mse:.2f}")

R² Score : 0.9927
RMSE : 5.44
MAE : 2.37
MSE : 29.58

[9]: model_package= {'model': model,'scaler': scaler,'label_encoders': label_reg,'features': features}
joblib.dump(model_package, 'svm_reg_model.pkl')
print("Sauvegarde : svm_reg_model.pkl")

Sauvegarde : svm_reg_model.pkl
```

## XGBoost Classification

```
[8]: model = XGBClassifier( random_state=42, n_estimators=100,max_depth=5,learning_rate=0.1,eval_metric="logloss")
model.fit(X_train_scaled, y_train)
print("Modele entraîné avec succès !")

Modele entraîné avec succès !

[9]: y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy : {accuracy * 100:.2f}%")
print(f"precision: {precision_score(y_test, y_pred)* 100:.4f}%")
print(f"f1_score: {f1_score(y_test, y_pred)* 100:.4f}%")
print(f"recall: {recall_score(y_test, y_pred)* 100:.4f}%")

Accuracy : 79.89%
precision: 77.0370%
f1_score: 73.7589%
recall: 70.7483%
```

```
[10]: package = {'model': model,'scaler': scaler,'label_gender': le_gender,'label_smokes': le_smokes,'feature_columns': features}
joblib.dump(package, "xgboost_class_model.pkl")
print("Sauvegarde terminée : xgboost_class_model.pkl")

Sauvegarde terminée : xgboost_class_model.pkl
```

## XGBOOST REGRESSION

```
[7]: model = XGBRegressor(random_state=42,n_estimators=100,max_depth=5,learning_rate=0.1)
model.fit(X_train_scaled, y_train)
print("Modele entraîné !")

Modele entraîné !

[8]: y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"R² Score : {r2:.4f} ")
print(f"RMSE : {rmse:.2f}")
print(f"MAE : {mae:.2f}")
print(f"MSE : {mse:.2f}")

R² Score : 0.9989
RMSE : 2.07
MAE : 1.47
MSE : 4.28
```

```
[9]: model_package = {'model': model,'scaler': scaler,'label_encoders': label_reg,'feature_columns': features}
joblib.dump(model_package, 'xgboost_reg_model.pkl')
print("Sauvegarde du modèle : xgboost_reg_model.pkl")

Sauvegarde du modèle : xgboost_reg_model.pkl
```

## Conclusion :

Ce projet nous a permis de créer une plateforme de Machine Learning fonctionnelle, facile à utiliser et capable de traiter différents jeux de données. Grâce à ce travail, nous avons appris à préparer les données, entraîner des modèles, évaluer leurs performances et les intégrer dans une application web.