



ARTIFICIAL INTELLIGENCE

PROJECT

PRÉDICTION DE LA POPULARITÉ MUSICALE &

DÉTECTION DU DIABÈTE VIA LA RÉGRESSION

LOGISTIQUE

RÉALISÉE PAR: RHAZI CHAIMAE

ENCADRÉ PAR: ADIL EL MAKRANI





TABLE OF CONTENTS

- Introduction 01
- Présentation des jeux de données 02
- Prétraitement (Spotify) 03
- Prétraitement (Diabetes) 04
- Analyse exploratoire 05
- Entraînement et validation (Spotify) 06





TABLE OF CONTENTS

• Entraînement et validation (Diabetes)	07
• Comparaison des deux modèles	08
• Prédiction (Spotify + Diabetes)	09
• Limites du projet	10
• Perspectives d'amélioration	11
• Conclusion générale	12



INTRODUCTION

Projet de classification supervisée appliquée à deux domaines :

- 🎵 Spotify : prédire si une chanson est populaire
 - ⌚ Diabète : prédire si un patient est diabétique
-

Objectif :

Construire, évaluer et comparer deux modèles de régression logistique sur des jeux de données réels.



PRÉSENTATION DES JEUX DE DONNÉES



Dataset Spotify

- 170 000+ chansons (1921-2020)
- Caractéristiques audio : valence, energy, danceability, tempo, loudness, etc.
- Cible : popularity_target (1 = populaire, 0 = non populaire)

Dataset Diabète :

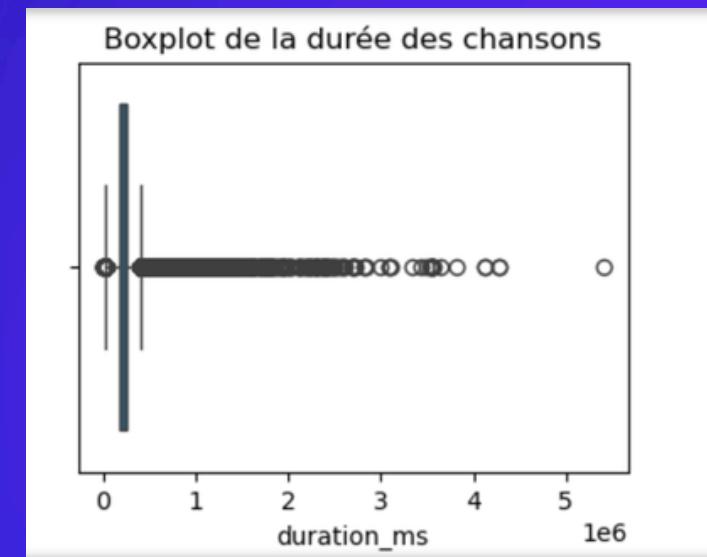
- >768 patients
- >Variables biométriques : glucose, bmi, âge, pression artérielle, etc.
- >Cible : Outcome (1 = diabétique, 0 = non diabétique)

PRÉTRAITEMENT (SPOTIFY)

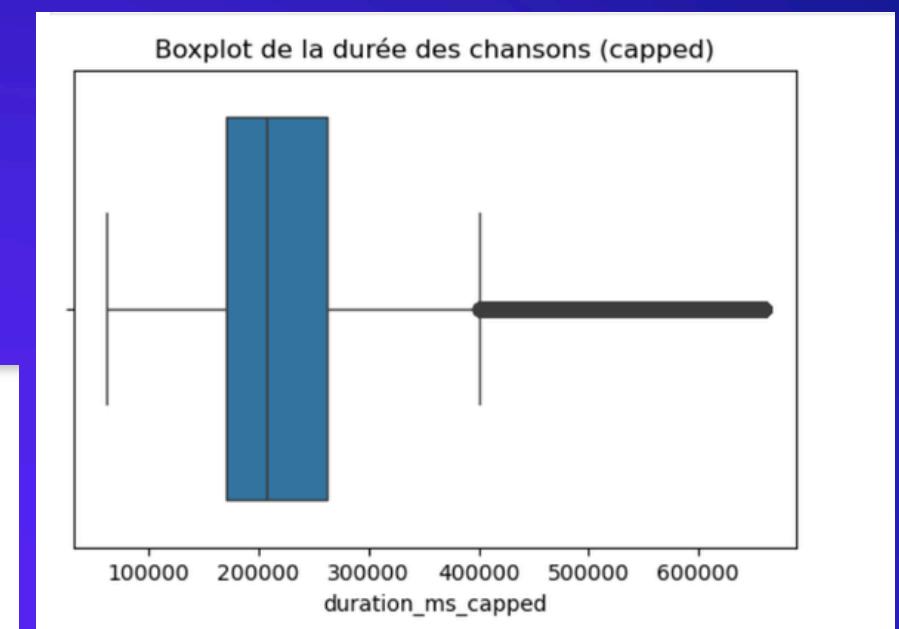
01

NETTOYAGE DES DONNÉES :

- ✓ SUPPRESSION DES DOUBLONS
- ✓ GESTION DES VALEURS MANQUANTES
- ✓ CORRECTION DES TYPES DE DONNÉES
- ✓ SUPPRESSION DES COLONNES INUTILES



Avant Capping



Après capping

02

TRAITEMENT DES VALEURS ABERRANTES :

- ✓ APPLICATION DE LA MÉTHODE DU CAPPING POUR LIMITER L'IMPACT DES EXTRÊMES

PRÉTRAITEMENT DES DONNÉES (SPOTIFY)



NORMALISATION :

✓ UTILISATION DE STANDARDSCALER
POUR METTRE TOUTES LES VARIABLES À
LA MÊME ÉCHELLE

valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness
0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...']	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOlz	0.878000
0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000
0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...']	0.328	500062	0.166	0	1o6l8BglA6ylDMrlELygv1	0.913000
0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028
0.2530	1921	0.957	['Phil Regan']	0.418	166693	0.193	0	4d6HGyGT8e121BsdKmw9v6	0.000002

EXTRAIT DES DONNÉES AVANT NORMALISATION

valence	acousticness	danceability	duration_ms_capped	year	energy	instrumentalness	liveness	loudness_capped
-1.786737	1.280973	-1.470667	4.454459	-2.160485	-1.018058	2.281156	2.625104	-1.537246
1.649666	0.615525	1.597303	-0.488312	-2.160485	-0.531898	-0.530427	-0.262500	-0.178292
-1.862797	1.225076	-1.192277	2.813666	-2.160485	-1.186344	2.393235	-0.599863	-0.605950
-1.385139	1.241046	-1.493392	-0.183803	-2.160485	-0.651568	-0.530338	1.001184	0.376474
-1.050473	1.214428	-0.680949	-0.631333	-2.160485	-1.085372	-0.530421	0.132044	0.238005

EXTRAIT DES DONNÉES APRÈS NORMALISATION

PRÉTRAITEMENT DES DONNÉES (SPOTIFY)

04

TRANSFORMATION DE LA CIBLE :

✓ CRÉATION DE LA VARIABLE BINAIRE POPULARITY_TARGET (BASÉE SUR
LA MÉDIANE DE POPULARITY)

```
print(df['popularity'].median())
34.0

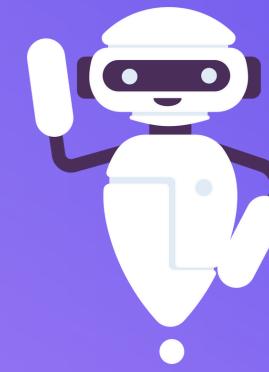
df_scaled['popularity_target'] = df['popularity'].apply(lambda x: 1 if x >= 34.0 else 0)
df_scaled.head()
```

popularity	popularity_target
4	0.0
5	0.0
5	0.0
3	0.0
2	0.0

```
print(df_scaled['popularity_target'].value_counts())
popularity_target
0.0    84785
1.0    84608
Name: count, dtype: int64
```

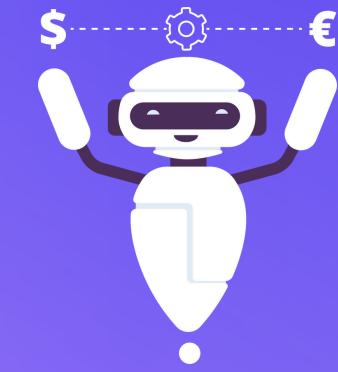
NOMBRES DES 1 ET 0 SONT BIEN EQUILIBRÉS

PRÉTRAITEMENT (DIABETES)



NETTOYAGE DES DONNÉES :

- ✓ SUPPRESSION DES DOUBLONS
- ✓ GESTION DES VALEURS MANQUANTES
- ✓ CORRECTION DES TYPES DE DONNÉES



REEMPLACEMENT DES ZÉROS NON
PLAUSIBLES (EX : GLUCOSE,
BMI, INSULIN) PAR LA MÉDIANE
DE CHAQUE COLONNE

```
(df_diabetes == 0).sum()
```

Pregnancies	111
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	500
dtype: int64	

COMBIEN DE 0 PAR COLONNE

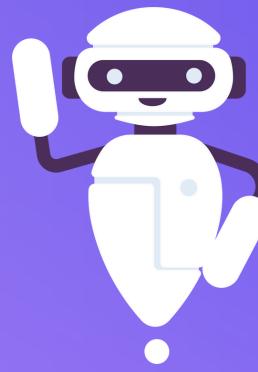
```
# Liste des colonnes où la valeur 0 n'est pas biologiquement plausible
cols_with_zeros = [ 'Glucose', 'BloodPressure', 'BMI', 'SkinThickness', 'Insulin' ]
# Je remplace les 0 par la médiane pour chaque colonne concernée
for col in cols_with_zeros :
    # Je calcule la médiane uniquement sur les colonnes différentes de 0
    median = df_diabetes[col][df_diabetes[col] != 0 ].median()
    # Je remplace les 0 par cette médiane
    df_diabetes[col] = df_diabetes[col].replace(0, median)

print((df_diabetes[cols_with_zeros] == 0).sum())
```

Glucose	0
BloodPressure	0
BMI	0
SkinThickness	0
Insulin	0
dtype: int64	

APRÈS REMplacement

PRÉTRAITEMENT (DIABETES)



NORMALISATION DES
VARIABLES NUMÉRIQUES AVEC
STANDARDSCALER POUR LES
METTRE SUR UNE MÊME
ÉCHELLE.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6		0.627	50
1	1	85	66	29	0	26.6		0.351	31
2	8	183	64	0	0	23.3		0.672	32
3	1	89	66	23	94	28.1		0.167	21
4	0	137	40	35	168	43.1		2.288	33

DONNÉES AVANT NORMALISATION



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.639947	0.866045	-0.031990	0.670643	-0.181541	0.166619		0.468492	1.425995
1	-0.844885	-1.205066	-0.528319	-0.012301	-0.181541	-0.852200		-0.365061	-0.190672
2	1.233880	2.016662	-0.693761	-0.012301	-0.181541	-1.332500		0.604397	-0.105584
3	-0.844885	-1.073567	-0.528319	-0.695245	-0.540642	-0.633881		-0.920763	-1.041549
4	-1.141852	0.504422	-2.679076	0.670643	0.316566	1.549303		5.484909	-0.020496

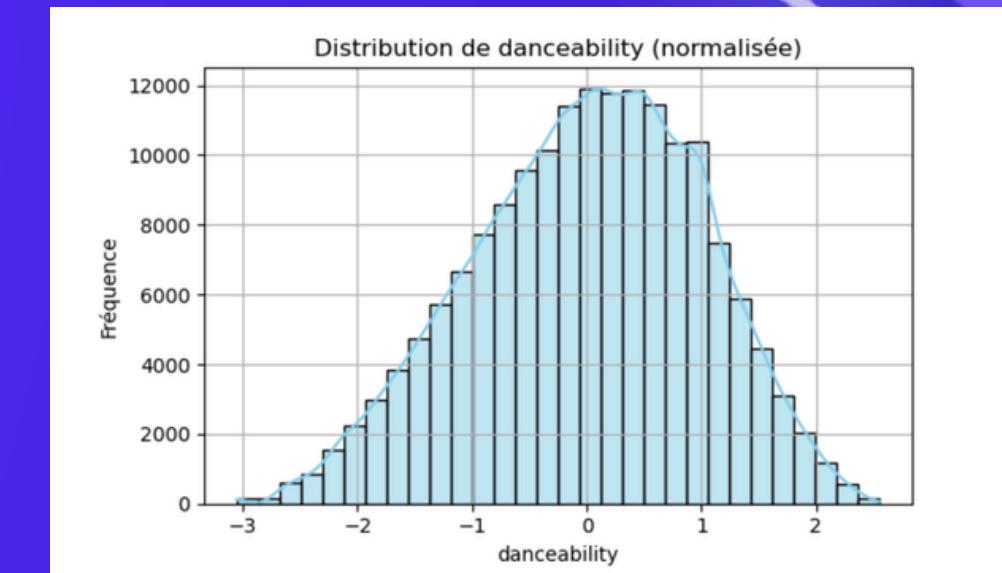
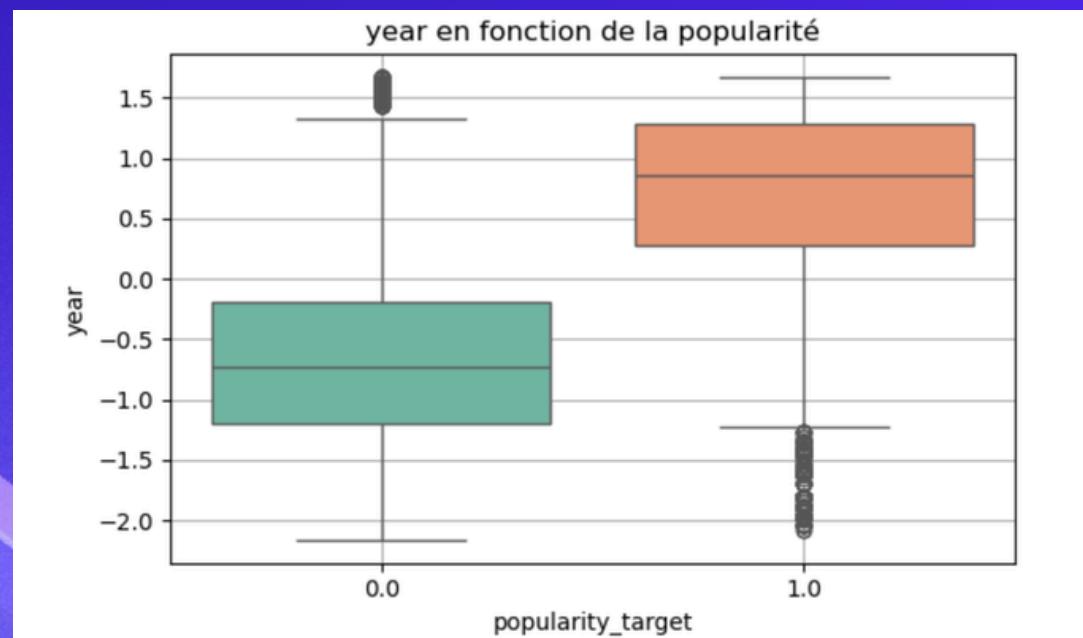
DONNÉES APRÈS NORMALISATION

ANALYSE EXPLORATOIRE



Histogrammes et boxplots pour analyser la distribution de chaque feature.

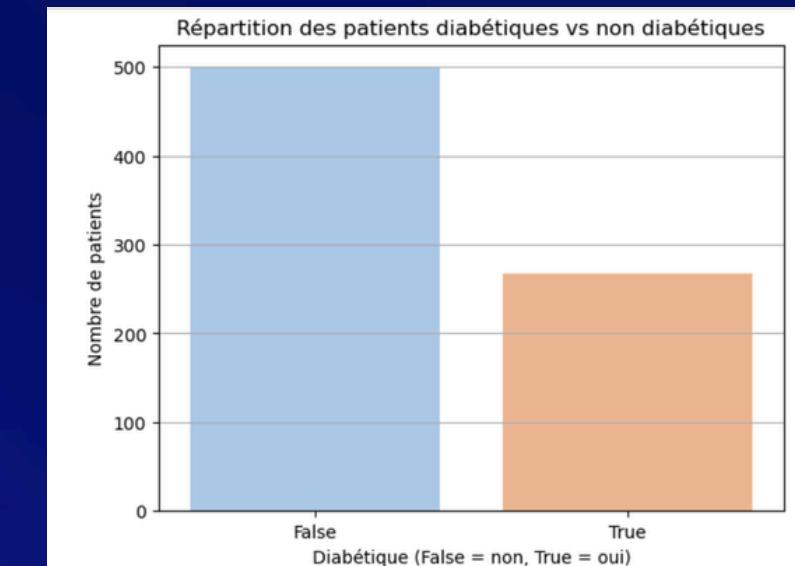
Corrélation forte observée entre year et popularity_target.



->Par exemple, la variable danceability présente une distribution légèrement centrée autour de 0.5, indiquant une tendance générale à produire des morceaux moyennement dansants.

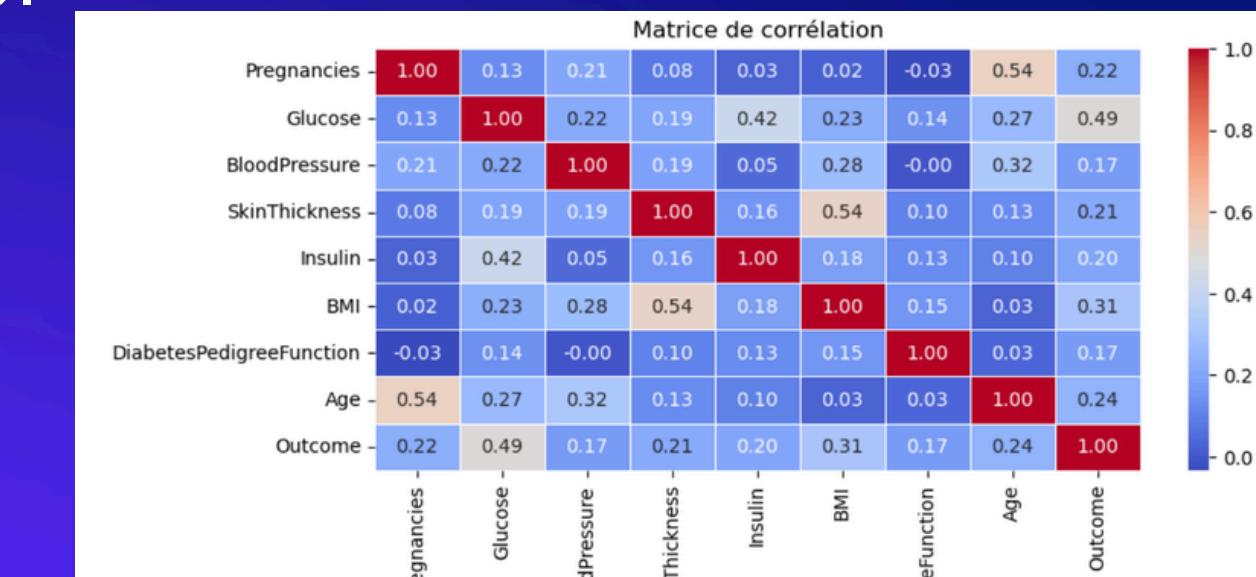
Les morceaux récents ont tendance à être plus populaires.

Distribution déséquilibrée de la variable Outcome (true \neq false).



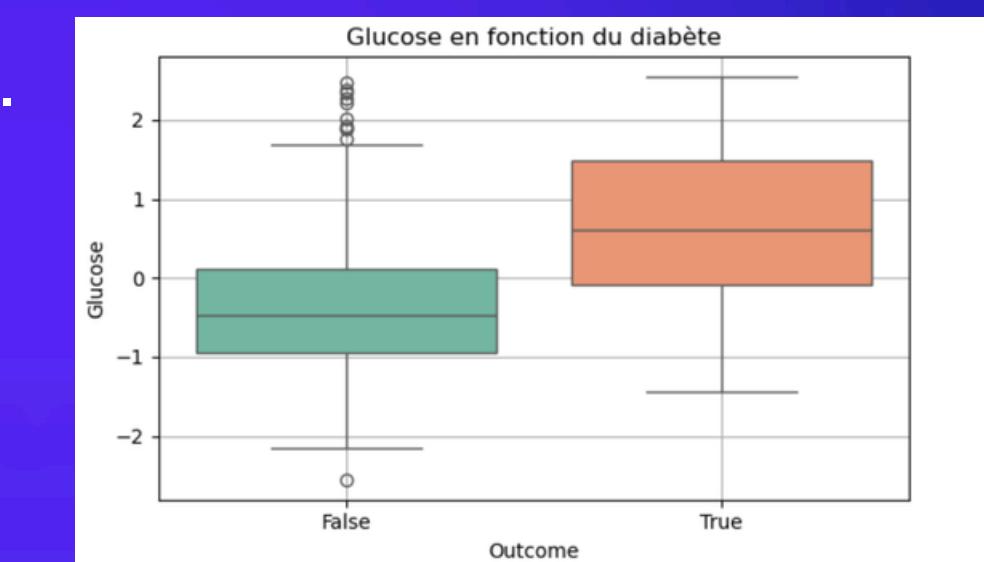
Corrélations notables :

Glucose, BMI et Age corrélés avec le diabète.



Visualisation via boxplots selon Outcome pour comprendre les différences entre patients diabétiques et non diabétiques.

ex: Glucose



ENTRAÎNEMENT ET VALIDATION (SPOTIFY)

01

Méthodologie

```
# Séparation des données en ensembles d'entraînement et de test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Séparation du dataset : 80 % entraînement / 20 % test

02

Modèle utilisé

```
# Créeation du modèle de régression Logistique  
model = LogisticRegression()
```

Régression logistique (modèle linéaire simple et interprétable)

Pas besoin de rééquilibrage (classes déjà bien réparties)

ENTRAÎNEMENT ET VALIDATION (SPOTIFY)

03

Entraînement réalisé sur plus de 135 000 chansons

Sauvegarde du modèle avec joblib pour une utilisation future

```
# Entraînement du modèle
model.fit(X_train, y_train)

▼ LogisticRegression
LogisticRegression()

import joblib

# Sauvegarder les objets entraînés
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(model, 'logistic_model.pkl')

print("Sauvegarde réussie.")
```

04

Résultats (sur le jeu de test)

Accuracy : 83 %

F1-score équilibré entre les classes

Bonne capacité à prédire les chansons populaires

```
Accuracy: 0.8256
Classification Report:
[[14026 2980]
 [ 2927 13946]]
             precision    recall   f1-score   support
          0.0       0.83      0.82      0.83     17006
          1.0       0.82      0.83      0.83     16873
   accuracy                           0.83      33879
  macro avg       0.83      0.83      0.83      33879
weighted avg       0.83      0.83      0.83      33879
```

ENTRAÎNEMENT ET VALIDATION (DIABETES)

Méthodologie

01

Séparation du dataset : 80 % entraînement / 20 % test

Stratification appliquée pour équilibrer les classes (false : non diabétique / true : diabétique)



Problème initial

Déséquilibre des classes : patients non diabétiques plus nombreux

Solution : SMOTE utilisé pour suréchantillonner la classe minoritaire

```
# Séparation des données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(
    X_res, y_res, test_size=0.2, random_state=42, stratify=y_res
)
```

ENTRAÎNEMENT ET VALIDATION (DIABETES)

02

Modèle

Régression logistique avec `class_weight='balanced'`

Entraînement effectué sur les données rééquilibrées

03

Résultats

Accuracy : 73.5 %

F1-score (classe True) : 0.73 (amélioré par rapport à avant SMOTE)

Matrice de confusion équilibrée

		precision			recall	f1-score	support
		False	0.73	0.75	0.74	100	
		True	0.74	0.72	0.73	100	
		accuracy			0.73	200	
		macro avg	0.74	0.73	0.73	200	
		weighted avg	0.74	0.73	0.73	200	



COMPARAISON DES DEUX MODÈLES

Analyse comparative :

- Le modèle **Spotify** performe mieux grâce à :
 - Un dataset plus grand et mieux équilibré
 - Des caractéristiques audio fortement corrélées à la popularité (ex. : year, energy)
- Le modèle **Diabète** montre plus de difficultés :
 - Problème médical plus complexe
 - Classe positive (patients diabétiques) plus difficile à détecter
 - Amélioration notable après application de **SMOTE**

Prédiction (Spotify + Diabète)

Speaker icon Spotify : prédire la popularité d'une nouvelle chanson

Entrée manuelle des caractéristiques audio d'une chanson (valence, energy, tempo, etc.)

Application du même prétraitement qu'à l'entraînement (capping + normalisation)

Le modèle prédit :

1 si la chanson a des chances d'être populaire

0 sinon

Affichage de la probabilité estimée de popularité



Diabète : prédire le risque pour un patient

```
# Prediction sur un nouveau patient
```

```
manual_input = [[  
    2, 130, 70, 28, 120, 31.5, 0.4, 30  
]]
```

Résultat prédiction : Non diabétique
Probabilité : 41.55%

```
# Prediction sur une nouvelle chanson  
manual_input = [[  
    0.125,      # valence faible  
    0.988,      # acousticness élevé  
    0.29,       # danceability faible  
    831667,     # durée moyenne courte  
    2022,       # année un peu plus proche de 2020  
    0.0628,     # énergie faible  
    0.867,      # instrumentalness élevé  
    0.601,      # Liveness élevé  
    -35.624,    # Loudness faible  
    0.0366,     # speechiness élevée  
    10,         # key  
    77.783     # tempo lent  
]]
```

Prédiction : 1
Probabilité d'être populaire : 92.68 %

Entrée des données médicales (âge, IMC, glycémie, etc.)

Utilisation du même pipeline de normalisation

Le modèle retourne :

diabétique si le patient risque d'être diabétique

non diabétique sinon

Affichage de la probabilité d'être diabétique

LIMITES DU PROJET



Spotify



Biais temporel : le dataset ne contient que des chansons jusqu'à 2020, donc la prédiction pour des années futures (ex. : 2025) peut être moins fiable.



Caractéristiques audio uniquement : le modèle n'exploite pas d'autres facteurs importants (paroles, artiste, marketing...).



Diabète



Taille du dataset limitée : seulement 768 patients, ce qui peut limiter la généralisation du modèle.



Variables médicales restreintes : d'autres facteurs de risque (antécédents familiaux, activité physique...) ne sont pas inclus

PERSPECTIVES D'AMÉLIORATION

1. Enrichir les données : intégrer plus de variables (ex. : paroles, artiste, campagne marketing pour Spotify / antécédents familiaux pour le diabète).
2. Tester d'autres modèles : arbres de décision, forêts aléatoires, SVM, ou réseaux de neurones.
3. Optimisation avancée : grid search, cross-validation plus poussée, sélection de variables.
4. Déploiement réel : créer une API ou interface web pour prédire automatiquement la popularité ou le risque de diabète.

CONCLUSION GÉNÉRALE

Deux problématiques différentes, une approche commune : la classification supervisée (Regression logistique)

Des résultats encourageants, surtout pour Spotify grâce à un dataset riche.

Le projet démontre l'importance du prétraitement, de l'analyse exploratoire, et de l'évaluation rigoureuse.

Une base solide pour explorer des projets avancés en Big Data appliquée.

THANK YOU!

