Université Ibn Tofaïl

Mini projet

Module

Intelligence Artificielle (ML & DL)

Du Rythme au Diagnostic : Deux Cas de Classification en Machine Learning

Année Universitaire 2025-2026

RÉALISÉE PAR:

RHAZI CHAIMAE

ENCADRÉ PAR:

ADIL EL MAKRANI

Remerciements

Je tiens à remercier chaleureusement toutes les personnes qui m'ont soutenu durant la réalisation de ce projet.

Je remercie en particulier notre professeur ADIL EL MAKRANI pour la qualité des cours dispensés, qui m'ont permis d'acquérir les connaissances nécessaires à la compréhension et à l'implémentation des techniques de traitement et d'analyse de données.

Je remercie également mes camarades pour leurs échanges et leur soutien, ainsi que ma famille pour son accompagnement moral tout au long de ce travail

Résumé

Ce projet a pour objectif de mettre en pratique les techniques d'analyse de données à travers deux études de cas distinctes. La première porte sur la prédiction de la popularité des chansons à partir de leurs caractéristiques audio extraites d'un dataset Spotify. La seconde concerne la prédiction de la présence ou non du diabète chez un patient à partir de données médicales.

Les étapes classiques du data science pipeline ont été suivies : nettoyage, traitement des valeurs manquantes, normalisation, visualisation exploratoire (EDA), puis modélisation à l'aide d'algorithmes de machine learning. Les modèles utilisés ont été évalués à l'aide de métriques telles que la précision (accuracy) et le rapport de classification.

Ce travail montre l'importance de la qualité des données et du choix des modèles dans la réussite d'un projet de machine learning, que ce soit pour des données musicales ou médicales.

Table de matière

Figures

Introduction

Avec la montée en puissance de la science des données, les algorithmes d'apprentissage automatique sont devenus des outils puissants pour analyser et prédire des phénomènes complexes dans divers domaines. Ce rapport présente deux projets de classification utilisant un seul algorithme commun : la régression logistique.

- ➔ Le **premier projet** concerne la **prédiction de la popularité de chansons** sur Spotify, à partir de leurs caractéristiques audio. À partir d'un ensemble de données publiques contenant plus de 170 000 morceaux, l'objectif est de prédire si une chanson sera populaire (ou non) en se basant sur des variables telles que la valence, le tempo, l'énergie, la durée etc.

- ➔ Le **deuxième projet** s'inscrit dans un cadre médical : il s'agit de **prédire si une personne est atteinte de diabète**, à partir d'attributs médicaux comme la glycémie, l'indice de masse corporelle ou la pression artérielle, à l'aide du célèbre jeu de données **Pima Indians Diabetes.**

Dans ces deux études de cas, nous avons volontairement utilisé **le même modèle de régression logistique** afin de comparer ses performances et son adaptabilité à deux contextes très différents : **la musique** et **la santé**.

4. Présentation des deux projets

Dans cette section, je présente en détail les deux projets de classification que j'ai réalisés. Malgré des domaines d'application très différents **(musique et santé)**, les deux projets reposent sur un même objectif : prédire une variable binaire à l'aide de la régression logistique.

La méthodologie appliquée aux deux projets suit la même structure générale (prétraitement, nettoyage, normalisation, modélisation). Cependant, certaines étapes ont été adaptées selon les spécificités de chaque jeu de données.

4.1 Projet 1 : Prédiction de la popularité d'une chanson

Ce projet utilise un ensemble de données Spotify contenant plus de 170 000 chansons publiées entre 1921 et 2020. Chaque chanson est décrite par des caractéristiques audio telles que :

- valence : le caractère émotionnel de la chanson (positif ou triste),
- danceability : la capacité à danser sur la chanson,
- energy, tempo, duration_ms, etc.

L'objectif principal était de prédire si une chanson serait populaire (1) ou non (0).

J'ai commencé par un nettoyage approfondi des données : traitement des valeurs manquantes, suppression des doublons, correction des types et gestion des valeurs aberrantes. Ensuite, j'ai normalisé les variables numériques à l'aide de **StandardScaler** afin de les mettre sur la même échelle.

Après cette étape, j'ai créé une nouvelle variable binaire **popularity_target** à partir de la variable popularity, en fixant un seuil basé sur la médiane. Cela m'a permis de reformuler le problème en une tâche de classification binaire.

J'ai ensuite effectué une analyse exploratoire des données (EDA) pour visualiser les distributions et détecter d'éventuelles corrélations.

Enfin, j'ai appliqué un modèle de régression logistique pour prédire la popularité d'une chanson, et j'ai évalué les performances du modèle à l'aide de métriques telles que : accuracy, précision, rappel et f1-score.

4.2 Projet 2 : Détection du diabète chez un patient

Ce second projet s'inscrit dans le domaine médical. Il a pour objectif de prédire si une personne est atteinte de diabète, à partir de caractéristiques biométriques, en utilisant un modèle de régression logistique.

Le dataset utilisé est le **Pima Indians Diabetes Dataset**, qui contient des données médicales réelles sur 768 patientes. Chaque observation est décrite par des variables telles que :

➔ Le nombre de grossesses (Pregnancies), Le taux de glucose (Glucose), La pression artérielle (BloodPressure), L'épaisseur du pli cutané (SkinThickness), Le taux d'insuline (Insulin), L'indice de masse corporelle (BMI), L'historique familial (DiabetesPedigreeFunction), L'âge (Age).

Les étapes suivies :

- Nettoyage des données : remplacement des valeurs biologiquement incohérentes (zéros) par la médiane.
- Normalisation des variables numériques avec StandardScaler.
- Modélisation avec une régression logistique (seuil par défaut).
- Évaluation via accuracy, précision, rappel, f1-score et matrice de confusion.

5. Méthodologie commune

Cette section décrit les étapes méthodologiques appliquées dans les deux projets **(Spotify et Diabète)**, allant de la collecte des données jusqu'à leur préparation pour l'entraînement des modèles de classification.

5.1 Collecte des données

Dans le cadre de ces deux projets, les données ont été récupérées depuis des sources ouvertes :

- ➔ **Projet Spotify:** Le dataset a été extrait de la plateforme [Kaggle] https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-1921-2020-160k-tracks?select=data.csv en particulier le fichier **`data.csv`**, contenant **170 653 lignes** et **19 colonnes**.

- ➔ **Projet Diabète:** Le dataset provient également de Kaggle, https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data intitulé **'Pima Indians Diabetes Database'** (fichier : **`diabetes.csv`**), qui contient **768 lignes** et **9 colonnes**.
  Il s'agit de données médicales de patientes (âge, glycémie, IMC, pression artérielle, etc.), avec comme variable cible `Outcome`, indiquant si la patiente est atteinte de diabète (`1`) ou non (`0`).

Les deux fichiers sont au format CSV et ont été chargés à l'aide de la bibliothèque **Pandas** en Python, grâce à la fonction `pd.read_csv()`.

★ Exploration initiale

Après le chargement des données avec `pd.read_csv()`, une première exploration a été faite avec :

- `df.head()` pour visualiser les premières lignes
- `df.info()` pour voir les types et les valeurs manquantes
- `df.describe()` pour les statistiques globales

Cela a permis de mieux comprendre la structure des données.

```
# Preparation des données
df = pd.read_csv('data.csv')
```

Figure 1: Chargement du fichier `data.csv` (Spotify) dans un DataFrame `df`.

```
df.head()
```

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665	-20.096	1
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160	-12.441	1
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101	-14.850	1
3	0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5	0.381	-9.316	1

Figure 2: Première lignes du dataset 'data.csv'

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
```

Figure 3: Illustration des types et des valeurs manquantes

```
df.describe()
```

	valence	year	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness
count	170653.000000	170653.000000	170653.000000	170653.000000	1.706530e+05	170653.000000	170653.000000	170653.000000	170653.000000	170653.000000
mean	0.528587	1976.787241	0.502115	0.537396	2.309483e+05	0.482389	0.084575	0.167010	5.199844	0.205839
std	0.263171	25.917853	0.376032	0.176138	1.261184e+05	0.267646	0.278249	0.313475	3.515094	0.174805
min	0.000000	1921.000000	0.000000	0.000000	5.108000e+03	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.317000	1956.000000	0.102000	0.415000	1.698270e+05	0.255000	0.000000	0.000000	2.000000	0.098800
50%	0.540000	1977.000000	0.516000	0.548000	2.074670e+05	0.471000	0.000000	0.000216	5.000000	0.136000
75%	0.747000	1999.000000	0.893000	0.668000	2.624000e+05	0.703000	0.000000	0.102000	8.000000	0.261000
max	1.000000	2020.000000	0.996000	0.988000	5.403500e+06	1.000000	1.000000	1.000000	11.000000	1.000000

Figure 4: Statistiques globales du dataset 'data.csv'

```
# Preparation des données
df_diabetes = pd.read_csv('diabetes.csv')
```

Figure 5: Chargement du fichier `diabetes.csv` dans un DataFrame `df_diabetes`.

```
df_diabetes.head()
```

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

Figure 6: Premières lignes du dataset 'diabetes.csv'

```
df_diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Figure 7: Illustration des types et des valeurs manquantes

```
df_diabetes.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Figure 8: Statistiques globales du dataset 'diabetes.csv'

5.2 Prétraitement et nettoyage

Le prétraitement est une étape cruciale avant toute modélisation. Elle permet d'assurer que les données sont cohérentes, propres et prêtes à être utilisées par les modèles de machine learning.

5.2.1 Pour dataset Spotify:

Le nettoyage du jeu de données Spotify a suivi plusieurs étapes importantes afin de préparer les données pour la modélisation. Voici le déroulement détaillé :

➢ **Vérification des valeurs manquantes**

Une vérification initiale des valeurs manquantes a été effectuée avec **df.isnull().sum()**. Aucune valeur **NaN** n'a été détectée.

➢ **Suppression des doublons**

Avec la commande **df.duplicated().sum()**, aucune ligne dupliquée n'a été trouvée. Aucune suppression nécessaire.

➢ **Vérification et correction des types de colonnes**

1. La colonne **release_date** était de type **object**, ce qui n'est pas approprié pour une date.

2. Elle a été convertie en **datetime.**

3.Après conversion, elle présentait une très grande diversité de valeurs → cela la rendait difficile à exploiter.

```
# 3. Vérifier les types des colonnes
df['release_date'] = pd.to_datetime(df['release_date'], errors='coerce')
print(df['release_date'].isnull().sum())  # voir combien de NaT après conversion
```

```
119798
```

Figure 9: Nombre de NAT après conversion

4. Elle a été jugée inutile car redondante avec la colonne year, et a donc été supprimée.

5. Les colonnes **explicit** et **mode**, initialement de type **int**, ont été converties en booléen **(True/False).**

➢ **Traitement des valeurs aberrantes**

Des valeurs extrêmes ont été identifiées dans les colonnes suivantes :

➢ Duration_ms, loudness, tempo.

→ Pour ces colonnes, un capping (bornage) a été appliqué afin de limiter les valeurs extrêmes et améliorer la qualité de l'apprentissage.

```
# Capping des valeurs extrêmes (1er et 99e percentile)
q_low = df['duration_ms'].quantile(0.01)
q_high = df['duration_ms'].quantile(0.99)

df['duration_ms_capped'] = df['duration_ms'].clip(lower=q_low, upper=q_high)
```

Figure 10: Exemple de code capping pour la colonne duration_ms

Idem pour les autres colonnes.
Des nouvelles colonnes ont été créées après "capping" :

➢ Duration_ms_capped, loudness_capped, tempo_capped

Les colonnes originales ont ensuite été supprimées.

➢ **Uniformisation des colonnes catégoriques**

Exemple : la colonne **artists** a été nettoyée en :

➔ Convertissant les valeurs en minuscules.

➔ Supprimant les espaces superflus en début et fin de chaîne.

➢ **Suppression des colonnes inutiles**

Les colonnes **id** et **name** ont été supprimées car elles n'étaient pas pertinentes pour l'apprentissage automatique.

➢ **Normalisation des données**

Un StandardScaler a été appliqué aux colonnes numériques pour normaliser les valeurs et accélérer l'apprentissage :

Colonnes normalisées :
valence, acousticness, danceability, energy, instrumentalness, liveness, speechiness, duration_ms_capped, loudness_capped, tempo_capped.

NB: La colonne popularity (cible) n'a pas été normalisée.

```
# Normalisation avec StandardScaler
numeric_columns = [ 'valence', 'acousticness', 'danceability', 'duration_ms_capped', 'year',
                    'energy', 'instrumentalness', 'liveness', 'loudness_capped', 'speechiness', 'key',
                    'tempo_capped']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[numeric_columns])

df_scaled = pd.DataFrame(X_scaled, columns=numeric_columns)


df_scaled['popularity'] = df['popularity'].values

df_scaled.head()
```

Figure 11: Normalisation avec StandardScaler (Spotify dataset)

➢ **Transformation de la variable cible**

Une nouvelle colonne **popularity_target** a été créée à partir de la variable **popularity** :

➔ Si **popularity** > médiane → 1 (populaire)

➔ Sinon → 0 (non populaire)

Cette transformation permet d'avoir une cible binaire adaptée à une classification.

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

```
34.0
```

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

Figure 12: Transformation de la variable cible (popularity_target)

➢ **Nettoyage final**

Les lignes sans cible (**popularity_target**) ou non exploitables ont été supprimées.

Le dataset final est propre et prêt pour l'analyse exploratoire!!

5.2.2 Pour dataset Diabetes:

- **Vérification des valeurs manquantes :**
  En utilisant les fonctions classiques de Pandas, nous avons confirmé qu'il n'existait aucune valeur manquante dans le dataset.

- **Suppression des doublons :**
  Aucune donnée dupliquée n'a été détectée **(df.duplicated().sum() == 0).**

- **Traitement des valeurs incohérentes (zéros biologiquement impossibles) :**

Certaines colonnes comme **Glucose, BloodPressure, SkinThickness, Insulin et BMI** contiennent des valeurs nulles représentées par 0, ce qui est biologiquement non plausible (ex : un niveau de glucose à 0).

Pour résoudre cela :

1. Nous avons identifié les zéros dans ces colonnes.

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

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

Figure 13: Nombre de 0 par colonne

2. Puis nous avons remplacé ces zéros par la **médiane calculée sur les valeurs non nulles** de chaque colonne concernée.

```
# Liste des colonnes ou 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 differents 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)
```

Figure 14: Code qui remplace les 0 par la médiane de chaque colonne concernée

- **Transformation de la variable cible `Outcome` :**
  Bien qu'elle soit déjà binaire (0 ou 1), nous avons converti la colonne en type booléen (**True** pour diabétique, **False** sinon) pour plus de clarté.

- **Normalisation des variables explicatives :**
  À l'aide du StandardScaler, toutes les colonnes numériques (sauf **Outcome**) ont été normalisées afin de faciliter l'apprentissage du modèle.

```python
# Normalisation avec StandardScaler
numeric_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_diabetes[numeric_columns])

df_scaled = pd.DataFrame(X_scaled, columns=numeric_columns)


df_scaled['Outcome'] = df_diabetes['Outcome'].values

df_scaled.head()
```

Figure 15: Normalisation avec StandardScaler (Diabetes dataset)

5.3 Analyse exploratoire des données (EDA)

5.3.1 Pour dataset Spotify:

L'analyse exploratoire des données du dataset Spotify a permis de mieux comprendre les variables influençant la popularité d'une chanson.
Tout d'abord, des **histogrammes** ont été tracés pour visualiser la distribution des variables numériques comme **danceability, energy, valence, tempo_capped,** etc.
Par exemple, la variable **danceability** présente une distribution légèrement centrée autour de 0.5, indiquant une tendance générale à produire des morceaux moyennement dansants.

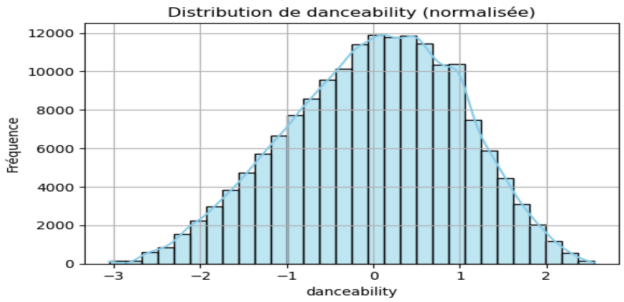

Figure 16 : Distribution de danceability

Ensuite, la variable cible **popularity_target** a été analysée à l'aide d'un graphique en barres.
La répartition montre que le dataset est **modérément équilibré** entre les chansons populaires (1) et non populaires (0), ce qui est favorable pour l'apprentissage du modèle de classification.

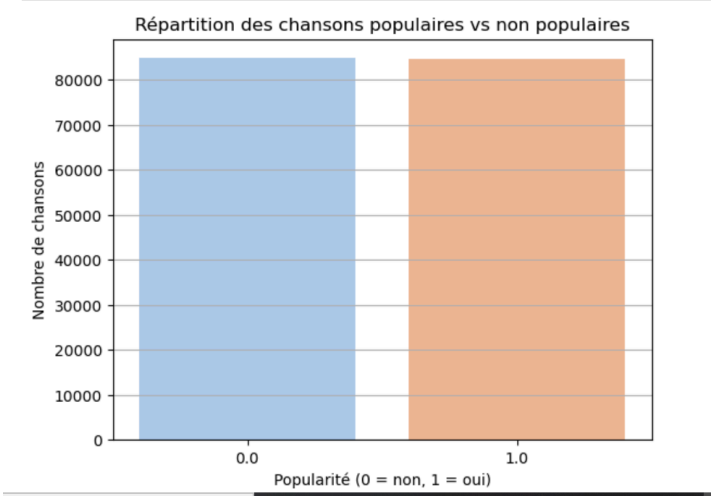

Figure 17: Répartition des chansons populaires vs non populaires

Une matrice de corrélation a été construite pour observer les relations entre variables.
Il a été observé que des variables comme **energy, loudness_capped** et **year** présentent une certaine corrélation positive entre elles, ce qui est cohérent puisque des morceaux énergiques sont souvent perçus comme plus "joyeux".

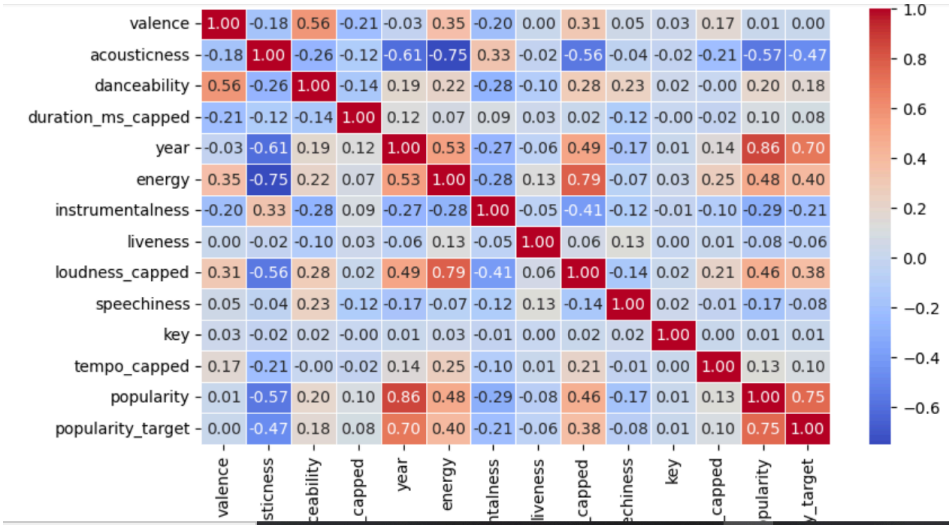

Figure 18: Matrice de corrélation dataset Spotify

Enfin, des boxplots ont été générés pour visualiser les variables selon la cible **popularity_target.**
Ils ont permis de confirmer la pertinence du **capping** appliqué à certaines variables comme **duration_ms_capped, tempo_capped** et **loudness_capped**, en réduisant l'impact des valeurs extrêmes.

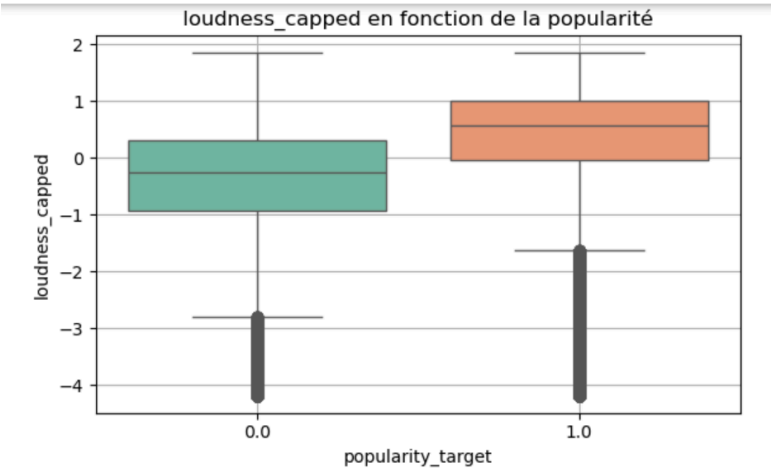

Figure 19: loudness_capped en fonction de la popularité

5.3.1 Pour dataset Diabetes:

Des **histogrammes** ont été générés pour les variables telles que glucose, BMI, insulin, age, etc.

La variable **glucose** présente une distribution **asymétrique à droite** (légèrement étalée vers les grandes valeurs), avec un **pic autour de la moyenne**. Cette forme suggère une concentration de patients autour d'un niveau de glucose normal, tandis qu'un plus petit nombre présente des valeurs élevées, ce qui est typique des patients atteints de diabète.

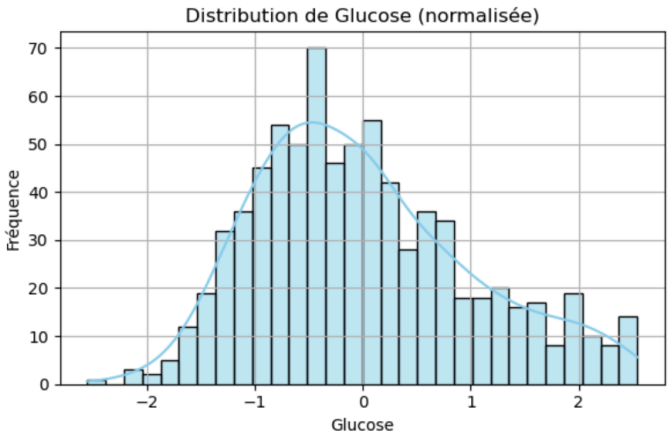

Figure 20: Distribution du Glucose

Ensuite, la variable cible **outcome** a été visualisée pour évaluer le déséquilibre de classes.
Le dataset présente un léger déséquilibre, avec plus de patients non diabétiques que diabétiques, ce qui devra être pris en compte lors du choix des métriques d'évaluation.

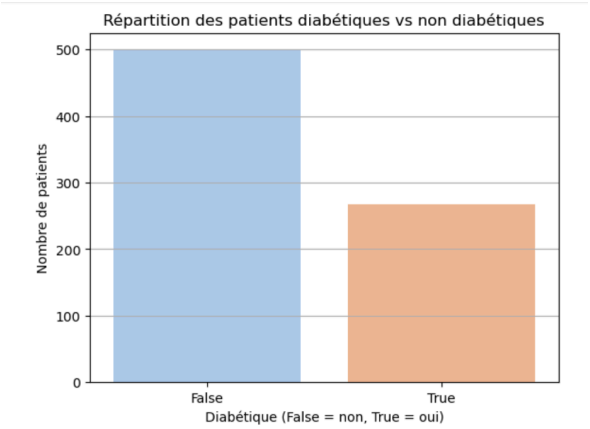

Figure 21: Répartition des patients diabétiques vs non diabétiques

Une matrice de corrélation a révélé une forte corrélation entre glucose, BMI et l'issue de la maladie (outcome), ce qui montre leur importance dans la prédiction.

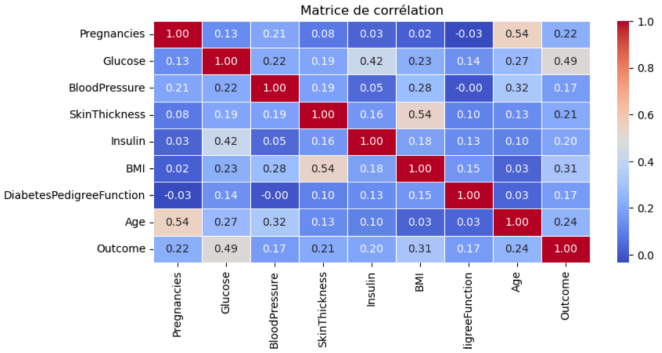

Figure 22: Matrice de corrélation dataset diabetes

Ensuite, des boxplots ont été tracés pour chaque variable numérique en fonction de la variable cible **Outcome.**

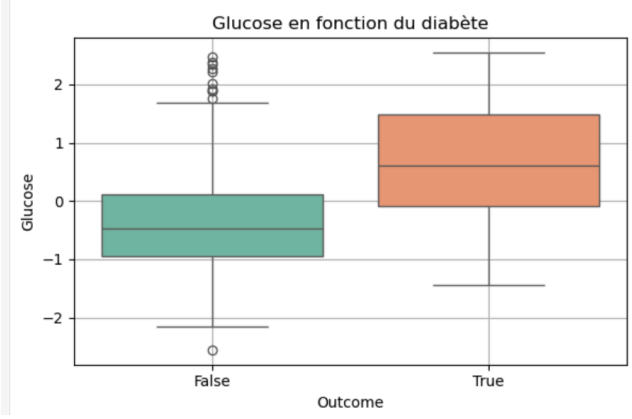

Figure 23 : Glucose en fonction du diabète

Le boxplot ci-dessus montre la répartition du taux de glucose en fonction de la présence ou non de diabète (Outcome). On constate que les patients diabétiques (True) ont des niveaux de glucose significativement plus élevés que ceux qui ne le sont pas (False). La médiane est nettement décalée vers le haut pour les diabétiques, ce qui corrobore le lien entre hyperglycémie et diabète.

6. Modélisation et évaluation

6.1 Choix du modèle:

Dans les deux projets, le modèle choisi est la **régression logistique**, qui est un algorithme de classification binaire.
Ce choix est motivé par sa simplicité, sa robustesse et sa capacité à fournir des probabilités interprétables.

- Dans le **projet Spotify**, il s'agit de prédire si une chanson est **populaire** (`1`) ou **non populaire** (`0`), à partir de variables numériques audio.

- Dans le **projet Diabète**, l'objectif est de déterminer si un patient est **atteint de diabète** (True) ou **non** (False), en se basant sur des paramètres médicaux.

Ce modèle est donc bien adapté aux deux cas, car la cible dans chaque dataset est **binaire**.

6.2 Entraînement et validation

Afin d'évaluer les performances de nos modèles de régression logistique, les données ont été divisées en deux sous-ensembles :

- **80 % des données** ont été utilisées pour l'**entraînement** du modèle,

- **20 % restantes** ont servi pour les **tests** et l'**évaluation**.

Cette séparation permet d'entraîner le modèle sur une large portion des données tout en préservant un ensemble indépendant pour juger objectivement sa performance.

➔ **Popularité des chansons**

Pour dataset Spotify, une séparation simple a été utilisée :

```
# 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)
```

Figure 24: Séparation des données en ensembles d'entraînement et de test

Le modèle a ensuite été entraîné avec une régression logistique standard, sans pondération particulière, étant donné que la variable cible **popularity_target** est modérément équilibrée.

```
# Création du modèle de régression logistique
model = LogisticRegression()
# Entraînement du modèle
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

Figure 25: Régression logistique

➔ **Détection du diabète**

Pour le dataset sur le diabète, une stratification a été appliquée :

```
# 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, stratify=y)
```

Figure 26: Séparation des données en ensembles d'entraînement et de test

Cela garantit une représentation équilibrée des classes (false : non diabétique, true : diabétique) dans les deux ensembles.
En complément, une pondération automatique des classes (**class_weight='balanced'**) a été ajoutée à la régression logistique pour compenser le léger déséquilibre des données.

```
# Création du modèle de régression logistique
model = LogisticRegression(class_weight='balanced')
# Entraînement du modèle
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(class_weight='balanced')
```

Figure 27: Régression logistique

Les modèles ainsi entraînés ont été sauvegardés à l'aide de la bibliothèque **joblib** pour permettre une réutilisation sans réentraînement !!

6.3 Rapport de classification

Une fois les modèles entraînés, une évaluation a été réalisée sur l'ensemble de test en utilisant plusieurs métriques classiques de classification binaire :

- Accuracy (exactitude) : pourcentage de prédictions correctes.
- Precision (précision) : proportion de vrais positifs parmi les prédictions positives.

- Recall (rappel) : proportion de vrais positifs parmi les cas réellement positifs.
- F1-score : moyenne harmonique entre la précision et le rappel.
- Matrice de confusion : tableau croisé des vraies classes et des prédictions.

➔ **Popularité des chansons**

Le modèle de régression logistique a donné des résultats satisfaisants :

- Accuracy : ~83 %
- F1-score (classe populaire) : élevé, indiquant une bonne capacité à identifier les chansons populaires.

```
# Prédiction sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluation du modèle
print(f'Accuracy: {accuracy_score(y_test, y_pred):.4f}')
print('Classification Report:')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
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
```

Figure 28: Résultats obtenues

- La matrice de confusion montre une répartition relativement équilibrée entre vrais positifs et vrais négatifs.

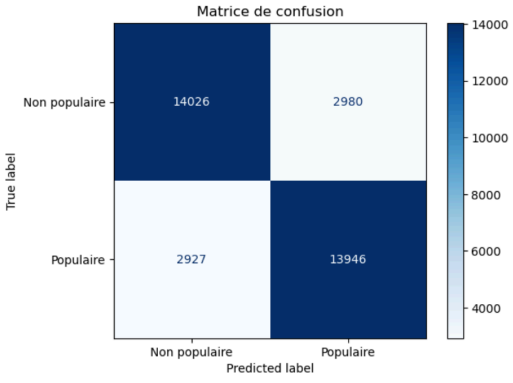

Figure 29: Matrice de confusion

➔ **Détection du diabète**

Le modèle de régression logistique appliqué au dataset du diabète a fourni des résultats acceptables :

- Accuracy : ~73 %, ce qui reste satisfaisant pour une première approche sur un jeu de données médical.
- F1-score pour la classe positive (patients diabétiques) : 0.65, ce qui montre que le modèle détecte modérément bien les cas positifs.

```
# Prédiction sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluation du modèle
print(f'Accuracy: {accuracy_score(y_test, y_pred):.4f}')
print('Classification Report:')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7338
Classification Report:
[[75 25]
 [16 38]]
              precision    recall  f1-score   support

       False       0.82      0.75      0.79       100
        True       0.60      0.70      0.65        54

    accuracy                           0.73       154
   macro avg       0.71      0.73      0.72       154
weighted avg       0.75      0.73      0.74       154
```

Figure 30: Résultats Obtenues

- La matrice de confusion indique que le modèle est plus performant pour identifier les patients non diabétiques (vrais négatifs) que pour détecter ceux atteints de diabète.

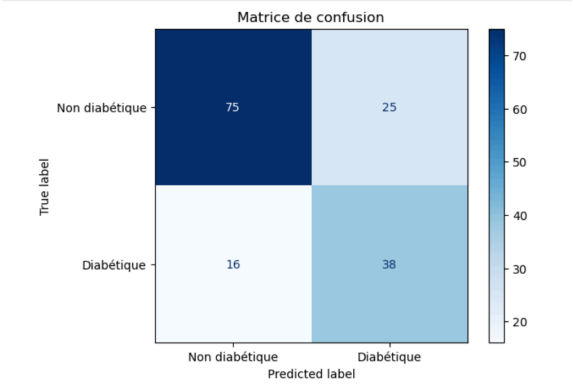

Figure 31: Matrice de confusion

Cela met en lumière un déséquilibre dans les performances entre les deux classes, typique des problèmes où la classe positive est plus difficile à capturer. Ce défi est courant en santé, où les cas positifs (ex. diabète) peuvent être plus rares ou plus complexes à prédire.

Ces résultats, bien qu'honnêtes pour un modèle linéaire simple, suggèrent des pistes d'amélioration, notamment :

- l'utilisation de méthodes de rééquilibrage (comme SMOTE, oversampling, ou ajustement des poids),
- l'adoption de modèles plus complexes (Random Forest, XGBoost),
- ou encore l'optimisation d'hyperparamètres.

★ Application de SMOTE pour l'équilibrage des classes

Afin d'améliorer la détection des cas positifs (patients atteints de diabète), la méthode **SMOTE (Synthetic Minority Oversampling Technique)** a été appliquée pour générer artificiellement de nouveaux exemples dans la classe minoritaire.

Cette technique a permis d'obtenir un **jeu de données équilibré** avant l'entraînement du modèle.

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)
```

```
# 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
)
```

Figure 32: Méthode SMOTE

Après l'entraînement du même modèle de régression logistique sur les données rééchantillonnées, les résultats ont été les suivants :

Accuracy : **73.5 %**

F1-score (classe True) : **0.73** (au lieu de **0.65** auparavant)

```
# Prédiction sur l'ensemble de test
y_pred = model.predict(X_test)

# Évaluation du modèle
print(f'Accuracy: {accuracy_score(y_test, y_pred):.4f}')
print('Classification Report:')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7350
Classification Report:
[[75 25]
 [28 72]]
              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
```

Figure 33: Résultats obtenues après SMOTE

Matrice de confusion équilibrée : bonne répartition entre les classes False et True.

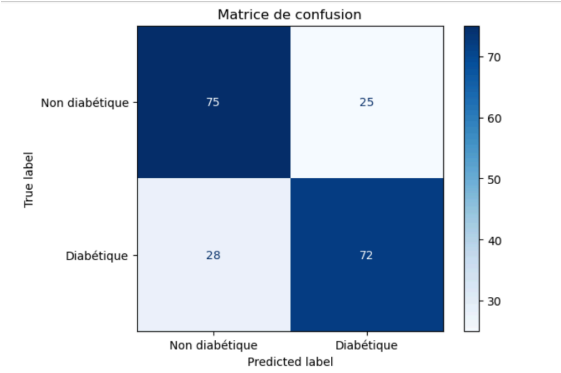

Figure 34: matrice de confusion après SMOTE

6.4 Prédiction

Prédiction sur une nouvelle chanson (Spotify)

Un exemple de chanson a été introduit avec les caractéristiques suivantes :

```
# 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
]]
# Appliquer le même prétraitement qu'en entraînement
manual_input_scaled_df = preprocess_new_song(manual_input)
```

```
prediction = model.predict(manual_input_scaled_df)
proba = model.predict_proba(manual_input_scaled_df)

print("Prédiction :", int(prediction[0])) # 1 pour populaire, 0 non populaire
print("Probabilité d'être populaire :", round(proba[0][1]*100, 2), "%")
```

```
Prédiction : 1
Probabilité d'être populaire : 92.68 %
```

Figure 35: Prédiction sur une nouvelle chanson

Cette chanson présente donc, selon le modèle, une forte probabilité d'être populaire, malgré certaines caractéristiques peu favorables comme la valence faible ou le tempo lent. Cela montre que d'autres facteurs peuvent compenser et influencer fortement la popularité. En effet, les morceaux plus récents (comme ici, 2022) sont généralement plus susceptibles d'être populaires selon les tendances observées dans le dataset.

Prédiction sur un nouveau patient (Diabetes)

Pour évaluer la capacité du modèle à généraliser, une prédiction a été réalisée sur un nouveau profil patient :

```
# Prediction sur un nouveau patient
manual_input = [[
    2, 130, 70, 28, 120, 31.5, 0.4, 30
]]
manual_input_df = pd.DataFrame(manual_input, columns=numeric_columns)

manual_input_scaled = scaler.transform(manual_input_df)
manual_input_scaled_df = pd.DataFrame(manual_input_scaled, columns=numeric_columns)
```

```
# Prédiction brute (0 ou 1)
prediction = model.predict(manual_input_scaled_df)[0]

# Probabilité que la patiente soit diabétique
proba = model.predict_proba(manual_input_scaled_df)[0][1]

print(f"Résultat prédiction : {'Diabétique' if prediction else 'Non diabétique'}")
print(f"Probabilité : {proba:.2%}")
```

```
Résultat prédiction : Non diabétique
Probabilité : 41.55%
```

Figure 36: Prédiction sur un nouveau patient

Ce résultat indique que, d'après le modèle, ce patient a une probabilité modérée d’être atteint de diabète.

7. Comparaison des performances des modèles

7.1 Précision, rappel, F1-score

Dans cette section, nous comparons les performances des deux modèles de régression logistique appliqués à deux jeux de données distincts :

- Projet 1 : Prédiction de la popularité d'une chanson (Spotify)
- Projet 2 : Détection du diabète chez un patient (Diabetes)

Modèle	Accuracy	Précision (classe positive)	Rappel (classe positive)	F1-score (classe positive)
Popularité des chansons	83 %	82 %	83 %	83 %
Détection du diabète	73 %	60 %	70 %	65 %

Figure 37: Comparaisons des performances des deux modèles

Analyse comparative

- Le modèle appliqué au dataset Spotify affiche de meilleures performances globales. Cela peut s'expliquer par :
    - La taille plus importante du dataset (plus de 170 000 chansons),
    - Des features audio bien corrélées à la cible, rendant la séparation entre classes plus aisée.
- Le modèle appliqué au dataset Diabetes montrait initialement des performances plus modestes, notamment au niveau de la précision (60 %).

  Toutefois, après application de la méthode SMOTE pour rééquilibrer les classes, les performances se sont significativement améliorées :

  - **Accuracy** : 73.5 %
  - **F1-score (classe True)** : 0.73 (au lieu de 0.65 auparavant)

  Ces résultats montrent que le **rééquilibrage des données a permis de mieux détecter les patients diabétiques**, en réduisant le déséquilibre de prédiction entre les deux classes.

Conclusion

- La régression logistique s’est révélée adaptée aux deux problèmes, mais avec des performances contrastées selon la nature du dataset.

8. Conclusion générale

Ce travail a permis de mettre en œuvre une démarche complète de classification supervisée à travers deux projets concrets :

- La prédiction de la popularité d'une chanson à partir de ses caractéristiques audio,
- Et la détection du diabète chez un patient à partir de variables biologiques.

Dans les deux cas, nous avons appliqué une méthode rigoureuse allant de la collecte et préparation des données à la modélisation, évaluation et interprétation des résultats.

Les résultats obtenus sont globalement satisfaisants.

- **Pour le dataset Spotify**, le modèle atteint une accuracy de 83 %, avec des métriques équilibrées entre les classes, montrant que les caractéristiques audio influencent efficacement la popularité.
- **Pour le dataset Diabète**, malgré une accuracy plus modeste (73 %), le modèle reste pertinent, tout en révélant les limites naturelles de la régression logistique sur des données médicales déséquilibrées.
- L'amélioration des performances du modèle pour le diabète grâce à l'application de SMOTE souligne l'importance des techniques de prétraitement dans les contextes de déséquilibre de classes, en particulier dans les domaines sensibles comme la santé.

Ce projet a permis d'expérimenter un cycle complet d'analyse prédictive, avec des données à volume significatif, dans une approche orientée Big Data.

9. Perspectives et améliorations possibles

Plusieurs pistes pourraient être envisagées pour améliorer la performance et approfondir le projet :

🎵 Projet 1 – Popularité d'une chanson :

- Intégrer des informations supplémentaires (comme le genre musical, l'artiste ou la date de sortie exacte).
- Tester d'autres modèles plus puissants : Random Forest, Gradient Boosting, etc.
- Réaliser une analyse temporelle de la popularité en fonction des années.

🩺 Projet 2 – Diabète :

- Poursuivre l'utilisation de techniques de rééquilibrage (comme SMOTE, déjà appliquée) et explorer d'autres approches comme l'undersampling ou l'ajustement de seuils de classification.
- Tester des modèles non linéaires plus complexes pour capturer les relations entre variables.
- Explorer des outils d'explicabilité (SHAP, LIME) pour mieux comprendre les prédictions.

Enfin, une généralisation de ce type de méthodologie à d'autres cas d'usage ouvrirait la voie vers des applications plus poussées en santé, musique, marketing, etc.

Références

Pandas Documentation – https://pandas.pydata.org/docs/
➤ Utilisée pour la manipulation des données (chargement, nettoyage, transformation…).

Matplotlib & Seaborn – https://matplotlib.org/ , https://seaborn.pydata.org/
➤ Pour la création de visualisations et d'analyses exploratoires

Imbalanced-learn Documentation (SMOTE)
https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
➤ Pour le rééquilibrage des classes dans le dataset Diabète.

Dataset Spotify Features – Kaggle
https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-1921-2020-160k-tracks?select=data.csv

Pima Indians Diabetes Dataset – UCI / Kaggle
https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/data
Pima Indians Diabetes Database