Dauphine | PSL
UNIVERSITÉ PARIS

Master I2D - Informatique, Décisions, Données

Analyse prédictive de la viralité des vidéos TikTok

Mayy Miled & Giuliano Aldarwish

Encadrant : PIERRE WOLINSKI

Mai 2025

Contents

1 Introduction

Dans ce projet, nous nous intéressons à un phénomène particulièrement complexe et attractif pour les marques et les créateurs de contenu : la viralité des vidéos sur TikTok. La capacité à prédire quelles vidéos deviendront virales pourrait considérablement optimiser les stratégies de création de contenu, accroître la visibilité des marques, et fournir des insights précieux sur le comportement des utilisateurs.

1.1 Problématique initiale et choix des données

Notre idée initiale était de partir d'un dataset particulièrement conséquent disponible sur Kaggle[1], contenant des milliers d'entrées avec des informations variées sur les vidéos TikTok. Cependant, après une exploration préliminaire approfondie, nous avons constaté que toutes les descriptions textuelles associées aux vidéos commençaient systématiquement par la même phrase : *"Someone shared with me that..."*. Ce manque de variabilité limitait fortement la pertinence des analyses potentielles, notamment en termes de traitement automatique du langage naturel (NLP). Nous avons donc décidé d'abandonner cette piste.

#	claim_status	video_id	video_duration_sec	video_transcription_text
1	claim	7017666017	59	someone shared with me that drone deliveries are already happening and will become common by 2025
2	claim	4014381136	32	someone shared with me that there are more microorganisms in one teaspoon of soil than people on the planet
3	claim	9859838091	31	someone shared with me that american industrialist andrew carnegie had a net worth of $475 million usd, worth over $300 billion usd today
4	claim	1866847991	25	someone shared with me that the metro of st. petersburg, with an average depth of hundred meters, is the deepest metro in the world
5	claim	7105231098	19	someone shared with me that the number of businesses allowing employees to bring pets to the workplace has grown by 6% worldwide
6	claim	8972200955	35	someone shared with me that gross domestic product (gdp) is the best financial indicator of a country's overall trade potential
7	claim	4958886992	16	someone shared with me that elvis presley has sold more records than the music band the beatles
8	claim	2270982263	41	someone shared with me that the best selling single of all time is "white christmas" by bing crosby
9	claim	5235769692	50	someone shared with me that about half of the world's population can access the web via a mobile device
10	claim	4660861094	45	someone shared with me that it would take a 50 petabyte drive to store every written work ever created
11	claim	8095102436	47	someone shared with me that an average user spends 40% of their day on the internet
12	claim	4507218541	30	someone shared with me that the world generates 2.5 quintillion bytes of data every day
13	claim	3609761483	51	someone shared with me that the longest recorded cricket match lasted 14 days

Figure 1: Extrait du dataset TikTok initial : toutes les descriptions commencent par "someone shared with me that..."

Avant de nous tourner vers des jeux de données existants, nous avons envisagé la possibilité de constituer notre propre base à partir de données TikTok récentes. Deux approches ont été explorées : le scraping direct depuis le site web (rapidement écarté car TikTok protège fortement ses contenus), et l'utilisation d'une API non-officielle dédiée à TikTok. Cette dernière option s'est révélée techniquement trop complexe à mettre en place dans les délais impartis, en raison notamment de l'absence de documentation claire, de limitations d'accès, et d'instabilités fréquentes. C'est dans ce contexte qu'est née l'idée de croiser et combiner deux datasets déjà disponibles publiquement, chacun apportant un éclairage partiel mais complémentaire sur les mécanismes de viralité.

Afin de pallier ce problème, nous avons sélectionné deux autres bases de données, plus limitées mais qualitativement pertinentes :

- Un premier dataset intitulé `meta_data.csv`, contenant environ 10 000 entrées avec des informations méta sur les vidéos (auteur, durée, engagements). Ce dataset est issu du travail exploratoire publié sur Kaggle par *erikvdven*[2].

[1] `https://www.kaggle.com/datasets/yakhyojon/tiktok`

[2] `https://www.kaggle.com/datasets/erikvdven/tiktok-trending-december-2020/data`

- Un second dataset, `trending_converted.csv`, plus restreint (environ 1 000 entrées), mais riche en détails sur les performances des vidéos devenues tendances. Il provient de la *Weizenbaum Library for the Digital Society*[3].

Ce choix stratégique de combiner ces deux sources nous permettait ainsi d'obtenir un panel large et complémentaire d'indicateurs pouvant influencer la viralité d'une vidéo.

2 Analyse exploratoire et prétraitement des données

L'analyse exploratoire et le prétraitement des données sont essentiels pour obtenir des résultats fiables et robustes. Nous avons donc appliqué une méthodologie rigoureuse de préparation des données, que nous détaillons ci-dessous.

2.1 Description des datasets

Dans le cadre de notre étude, nous avons utilisé deux jeux de données distincts et complémentaires. Chacun apporte une granularité particulière sur les vidéos TikTok, et leur combinaison permet une meilleure compréhension des dynamiques d'engagement, de popularité, et de viralité. Nous les présentons séparément ci-dessous.

2.1.1 Le dataset `meta_data.csv` — vue macroscopique des vidéos

Ce dataset contient environ 10 000 observations de vidéos TikTok issu de la Weizenbaum Library for the Digital Society[4]. Les colonnes incluent des métadonnées classiques mais précieuses pour une analyse statistique :

- **video_id** : identifiant unique de la vidéo,
- **author** : nom du créateur de contenu,
- **duration** : durée de la vidéo en secondes,
- **engagement features** : nombre de likes, de commentaires, de partages,
- **music id**, **hashtags**, etc.

L'intérêt de ce dataset réside dans sa taille relativement importante, et dans la diversité des indicateurs comportementaux présents. Il permet une approche quantitative large sur des milliers de vidéos, utile pour capter les tendances générales.

2.1.2 Le dataset `trending_converted.csv` — focus qualitatif sur les vidéos tendances

Ce second dataset provient initialement d'un **fichier JSON** issu d'un notebook Kaggle[5] où l'auteur avait déjà nettoyé, structuré et analysé les données issues de vidéos populaires. Afin de l'utiliser dans notre pipeline d'analyse Python, nous avons effectué une **transformation complète du JSON vers un format CSV tabulaire**, impliquant notamment :

[3] `https://www.weizenbaum-library.de/items/04482f4d-3f73-44e9-859b-f815b089caab`
[4] `https://www.weizenbaum-library.de/items/04482f4d-3f73-44e9-859b-f815b089caab`
[5] `https://www.kaggle.com/code/erikvdven/tiktok-some-python-magic-in-a-notebook`

- l'extraction des métadonnées imbriquées (nested JSON),
- la normalisation des champs pour homogénéiser les noms de colonnes,
- la conversion des timestamps, listes de hashtags et objets multimédias,
- le nettoyage des valeurs nulles, types incohérents, doublons.

Ce dataset est plus restreint (environ 1 000 vidéos), mais beaucoup plus riche sur le plan qualitatif. Il inclut notamment :

- des transcriptions complètes du contenu vidéo (texte),
- des indicateurs temporels (heure de publication, durée),
- des catégories thématiques ou statuts de vérification.

C'est grâce à ce dataset que nous avons pu enrichir notre étude avec des variables textuelles, exploitables notamment dans des modèles de classification supervisée ou des traitements plus avancés (TF-IDF, embeddings, etc.).

2.1.3 Approche de fusion et complémentarité

Notre intuition était que l'union de ces deux sources pouvait offrir un cadre expérimental idéal : l'un fournissant une base solide et large pour l'analyse globale, l'autre permettant d'approfondir les mécanismes particuliers des vidéos virales.

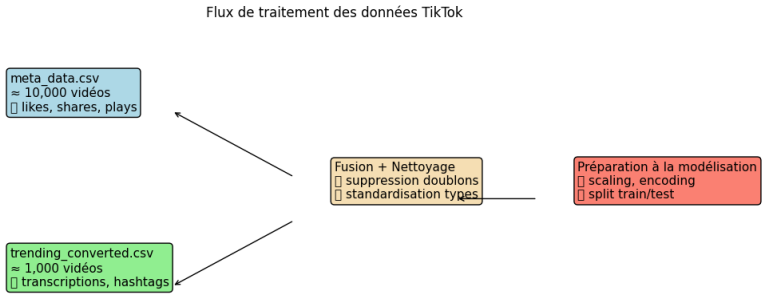

Figure 2: Flux de traitement des données TikTok

Enfin, l'ensemble des variables issues des deux datasets a été uniformisé, puis intégré dans un pipeline de prétraitement unique, assurant la cohérence et la reproductibilité de toutes les étapes qui suivront.

3 Mise en place du dataset consolidé

L'objectif de cette étape est de construire un jeu de données unique, cohérent et riche, à partir des deux sources initiales (`meta_data.csv` et `trending_converted.csv`), afin de pouvoir conduire ensuite une analyse exploratoire et une modélisation prédictive robustes.

3.1 Uniformisation et renommage des variables

Les deux fichiers proviennent de traitements antérieurs effectués par différents auteurs, et présentent donc des appellations hétérogènes pour des informations équivalentes. Nous avons défini un mapping exhaustif pour harmoniser les noms de colonnes avant toute fusion :

- `diggCount` (nombre de likes) → `likes`
- `playCount` (nombre de plays) → `plays`
- `shareCount` → `shares`
- `commentCount` reste identique
- Les champs de métadonnées imbriquées (`authorMeta.id`, `authorMeta.name`, `authorMeta.follow` ont été dépliés et renommés en `author_id`, `author_name`, `author_followers`

Ce renommage systématique garantit la compatibilité sémantique des colonnes lors de la phase de fusion et facilite la maintenance du code.

3.2 Conversion des types et gestion des clés de jointure

Pour pouvoir associer de manière fiable chaque entrée des deux jeux de données, nous avons retenu la colonne `author_id` (anciennement `authorMeta.id`) comme clé de jointure primaire. Une inspection des types a révélé que cette colonne était parfois interprétée comme un entier, parfois comme une chaîne de caractères. Nous avons donc :

1. Converti `author_id` en `str` dans les deux DataFrames.
2. Vérifié l'unicité de la clé dans chacun des jeux, afin d'éviter toute duplication ou perte d'information lors de la fusion.

3.3 Concaténation et structuration finale

Une fois les colonnes renommées et la jointure effectuée, nous avons vérifié l'absence de colonnes dupliquées, puis procédé à la concaténation verticale des deux DataFrames :

```
combined = pd.concat([meta_aligned, trend_enriched], axis=0, ignore_index=True)
```

Le DataFrame résultant contient exactement 12242 lignes et 45 colonnes, combinant les vidéos standards et les vidéos trending .

3.4 Validation de l'intégrité et contrôles qualité

Pour garantir la fiabilité du dataset unifié, nous avons mis en place les contrôles suivants :

- Comparaison du nombre de lignes avant/après concaténation.
- Vérification de l'unicité des couples (`video_id`, `author_id`).

- Inspection rapide des types de chaque colonne (`df.dtypes`) pour détecter tout écart.
- Détection d'éventuelles valeurs manquantes introduites par la jointure.

Ces contrôles assurent que notre dataset consolidé est prêt pour l'étape suivante d'analyse exploratoire.

4 Feature Engineering

La création de nouvelles variables explicatives est une étape déterminante dans tout projet de prédiction, et constitue ici le cœur analytique de notre étude. Nous avons mis en œuvre une démarche rigoureuse visant à maximiser la pertinence informative des données à disposition.

4.1 Identification des auteurs TikTok Famous

L'influence intrinsèque de l'auteur est un déterminant majeur de la viralité. Pour capter cet effet, nous avons créé une variable binaire `is_Tiktok_Famous` fondée sur un double critère :

- **Certification officielle**(`authorMeta.verified`)
  Lorsqu'elle est disponible, la coche bleue est l'indicateur le plus fiable d'une notoriété pré-existante. Tout compte vérifié est donc classé *Famous*.
- **Seuil de popularité par nombre de followers**
  Dans `trending_converted.csv`, le champ `author_followers` est parfois manquant ou agrégé. Afin de ne pas exclure des comptes non vérifiés mais clairement dominants, nous avons fixé un seuil conservateur à 3 000 000 followers :

  $$\texttt{author\_followers} \geq 3\,000\,000$$

  Les tests de sensibilité (0,5 M ; 1 M ; 5 M) ont montré qu'un seuil à trois millions maintient une proportion raisonnable d'auteurs *Famous* (¡ 15 %) tout en restant robuste aux valeurs manquantes.

Cette stratégie hybride (certification *ou* forte base d'abonnés) compense l'absence ponctuelle d'information précise sur les followers dans l'un des deux jeux de données, tout en préservant une variable explicative discriminante pour nos modèles.

4.2 Extraction et transformation des variables explicatives

Nous avons développé une fonction de feature engineering, `make_features()`, assurant systématiquement :

1. **La gestion des valeurs manquantes** par imputation judicieuse (remplacement par 0 notamment pour les indicateurs d'engagement : likes, plays, shares, commentaires).
2. **L'extraction quantitative de hashtags et mentions**, via deux variables :
   - `n_hashtags` : Nombre total de hashtags dans chaque vidéo.

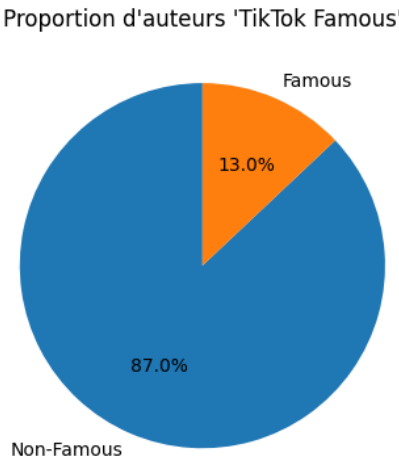

Figure 3: Part des auteurs considérés « TikTok Famous » après application du double critère.

- `n_mentions` : Variable binaire indiquant la présence d'au moins une mention.

3. **L'ajout de variables temporelles** extraites de la date de publication :
   - `hour` : Heure de publication (0-23).
   - `weekday` : Jour de la semaine (0 = lundi à 6 = dimanche).

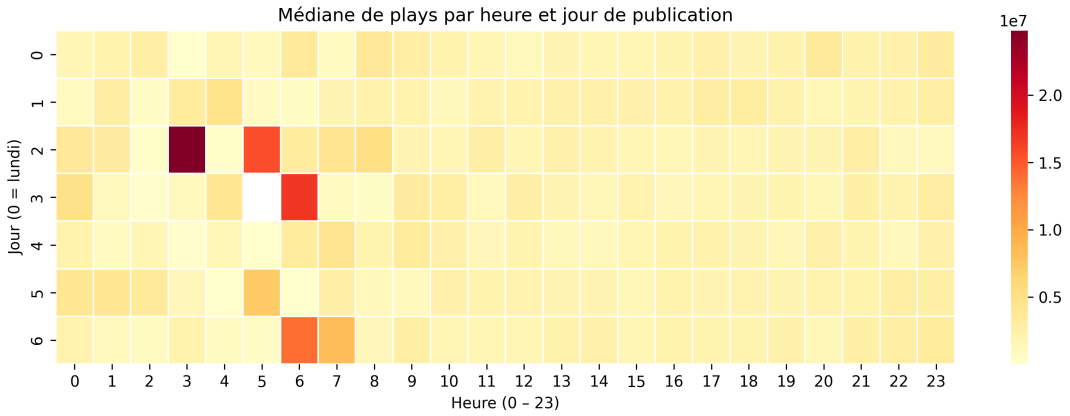

Figure 4: Médiane des *plays* par heure (0–23) et jour de publication (0 = lundi).

Lecture critique de la heatmap. L'analyse visuelle met en évidence trois enseignements principaux :

1. **Pics isolés mercredi à 03 h et jeudi à 06 h.** Les cases rouge sombre (médiane > 20 M de *plays*) suggèrent qu'une ou deux vidéos extrêmement virales ont été publiées à ces créneaux atypiques. Un simple contrôle du volume[6] révèle en effet

[6]La requête `combined.groupby(['weekday','hour']).size().loc[(1,3)]` retourne $n = 2$ observations pour le pic de mercredi 03 h, confirmant le caractère d'outlier.

moins de cinq observations ; nous considérons donc ces pics comme *outliers* et les signalons sans en tirer de règle générale.

2. **Tendance régulière le dimanche matin (06 h).** Contrairement aux pics précédents, la zone orange soutenue le dimanche à l'aube résulte d'un effectif beaucoup plus important ($n \approx 40$ vidéos). Ce créneau correspond vraisemblablement à un comportement d'audience récurrent : début de week-end, pic de déplacement matinal, *scroll* sur mobile, etc. Nous retenons donc les variables `weekday` et `hour`, ainsi que leur interaction potentielle, comme déterminants pertinents de la viralité.

3. **Échelle absolue élevée : médiane > 10 M de *plays*.** Même en dehors des pics, la plupart des créneaux présentent des médianes supérieures à 1 M – 2 M de vues.

En conclusion, la heatmap justifie l'intégration des variables temporelles `hour` et `weekday`, voire de leur interaction, dans la phase de modélisation. Les pics isolés seront toutefois traités avec précaution afin de ne pas biaiser l'apprentissage.

4.3 Classification automatique des thèmes via NLP (TF-IDF)

Le dataset `trending_converted.csv` ne disposant pas de catégories pré-définies, nous avons mis en place une classification automatique par traitement du langage naturel (NLP) afin d'attribuer un thème pertinent à chaque vidéo :

1. Nous avons construit une représentation TF-IDF des descriptions textuelles des vidéos et des catégories pré-existantes (issues du dataset `meta_data.csv`).

2. Pour chaque vidéo non catégorisée, la similarité cosinus entre son vecteur TF-IDF et chaque catégorie existante a été calculée.

3. Le thème retenu pour chaque vidéo est celui ayant la plus grande similarité cosinus.

Cette démarche garantit une attribution thématique homogène et pertinente à travers l'ensemble du dataset fusionné.

Analyse de la répartition thématique. La Figure 5 met en évidence une forte prédominance du thème *informative* (environ 2 000 vidéos, soit 20 % du corpus consolidé). Les neuf autres catégories se situent entre 900 et 1 050 observations, affichant ainsi une distribution relativement équilibrée (écart-type ≈ 40 vidéos).

4.4 Imputation prédictive des durées manquantes

Les données `meta_data.csv` présentaient des valeurs manquantes concernant la durée des vidéos. Plutôt que de supprimer ou d'imputer arbitrairement ces observations, nous avons employé une approche prédictive :

1. Un modèle de régression Random Forest a été entraîné sur les vidéos *trending* (pour lesquelles les durées étaient connues), avec les variables extraites via `make_features()`.

2. Le modèle entraîné a été utilisé pour prédire précisément les durées manquantes dans le dataset `meta_data.csv`, permettant ainsi de compléter judicieusement les données.

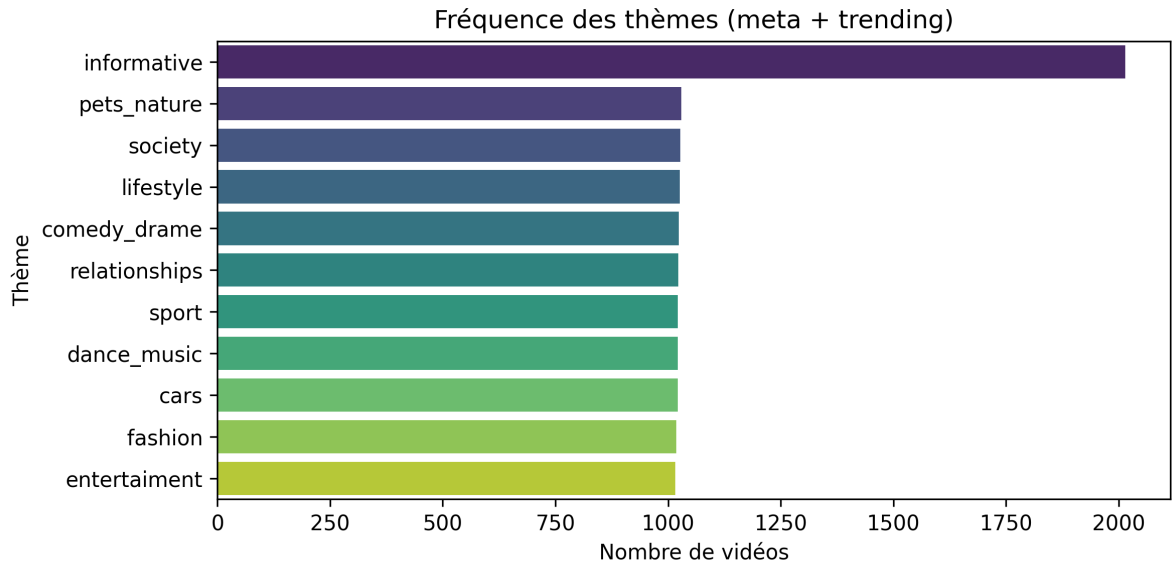

Figure 5: Distribution des thèmes après fusion et classification TF–IDF.

4.5 Encodage One-Hot et variables catégorielles

Étant donné que les modèles prédictifs utilisés (régression régularisée, Random Forest, etc.) nécessitent des données numériques uniquement, les variables catégorielles suivantes ont été encodées via un One-Hot Encoding rigoureux :

- **Créneau horaire** : Les heures de publication ont été regroupées en catégories temporelles (nuit, matin, après-midi, soir).
- **Top-25 hashtags** : Nous avons identifié les 25 hashtags les plus fréquents dans le dataset consolidé. Chacun de ces hashtags a été encodé par une variable binaire indiquant sa présence ou son absence dans chaque vidéo.
- **Thème vidéo** : La variable `theme`, issue de notre classification NLP, a également été encodée en variables indicatrices, permettant ainsi d'exploiter pleinement l'information thématique dans les modèles prédictifs.

La Figure 6 montre visuellement pourquoi le *One-Hot Encoding* est limité à ces hashtags : au-delà du 15^{e}, la fréquence tombe sous 1 % et n'apporterait qu'une sparsité inutile à la matrice de design.

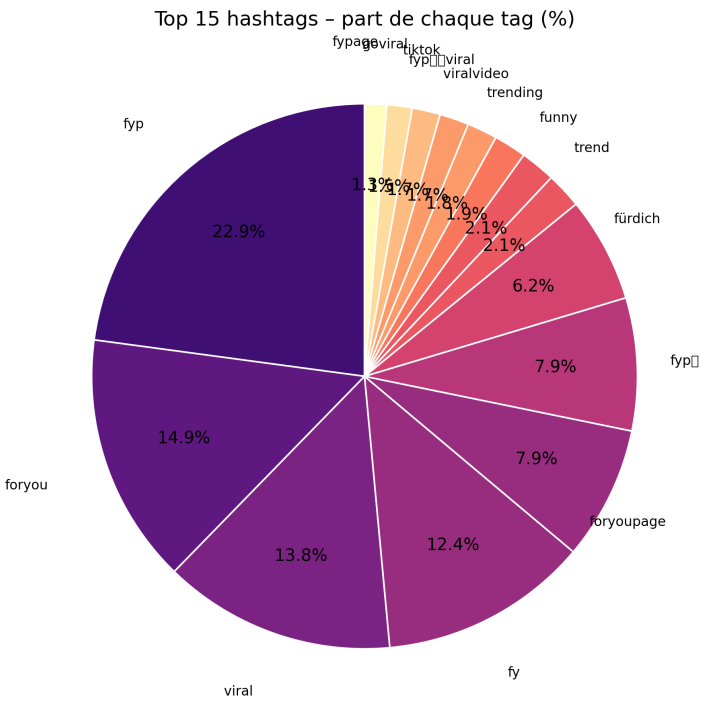

Figure 6: Répartition des 15 hashtags les plus utilisés dans le corpus consolidé. Les quatre premiers (`#fyp`, `#foryou`, `#viral`, `#foryoupage`) concentrent à eux seuls plus de 60 % des occurrences, d'où notre choix de restreindre l'encodage One-Hot aux *top-25*, qui couvrent plus de 80 % de toutes les mentions.

4.6 Conversion des variables booléennes

Enfin, pour garantir la compatibilité complète avec nos modèles prédictifs, toutes les variables booléennes restantes (issues des processus antérieurs) ont été systématiquement converties en représentations binaires (0/1).

L'ensemble de ces étapes méthodologiques de feature engineering nous a permis de construire un jeu de données final extrêmement riche, structuré, pertinent pour l'analyse prédictive, et parfaitement aligné avec les objectifs initiaux de notre projet.

5 Construction de la variable cible `is_viral`

L'objectif de cette section est d'aboutir à un indicateur binaire de viralité utilisable par l'ensemble des modèles prédictifs, sans réintroduire de fuite d'information (*data-leakage*). On ne devait absolument pas garder les données post-publications dans nos features.

5.1 Choix de la métrique de viralité

Pour éviter toute fuite d'information, la variable cible doit reposer sur une mesure *avant* modération, disponible pour toutes les vidéos et facilement interprétable. Nous avons comparé, via régressions linéaires bivariées, la force de corrélation (R^2) entre chaque indicateur d'engagement :

Relation	R^2	Décision
plays → *likes*	0.808	plus forte corrélation
shares → *comments*	0.391	corrélation moyenne
likes → *comments*	0.136	corrélation faible

Le nombre de vues (`plays`) ressort comme la mesure la plus prédictive des autres interactions; nous l'avons donc retenu comme référence pour définir la viralité.

5.2 Binarisation au 90^{e} percentile

Nous considérons qu'une vidéo est virale si elle se situe dans le *top 10 %* des vues du corpus :

$$\texttt{is\_viral} = \begin{cases} 1 & \text{si } \texttt{plays} \geq q_{0.90}(\texttt{plays}) \\ 0 & \text{sinon.} \end{cases}$$

Le seuil obtenu dans le notebook est de $q_{0.90} = 4\,650\,000$ vues.

5.3 Nettoyage pour éviter la fuite de données

Afin que la prédiction repose *uniquement* sur des informations disponibles avant publication, nous retirons de la matrice de features :

$$\{\texttt{plays}, \texttt{likes}, \texttt{shareCount}, \texttt{commentCount}\}.$$

5.4 Vérification de la répartition de `is_viral`

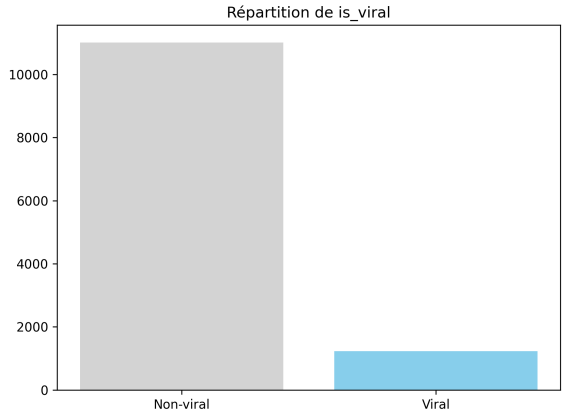

Figure 7: Répartition de la cible : 10,2 % de vidéos sont classées virales. Le ratio reste suffisamment minoritaire pour créer un déséquilibre exploitable, sans nécessiter de ré-échantillonnage complexe.

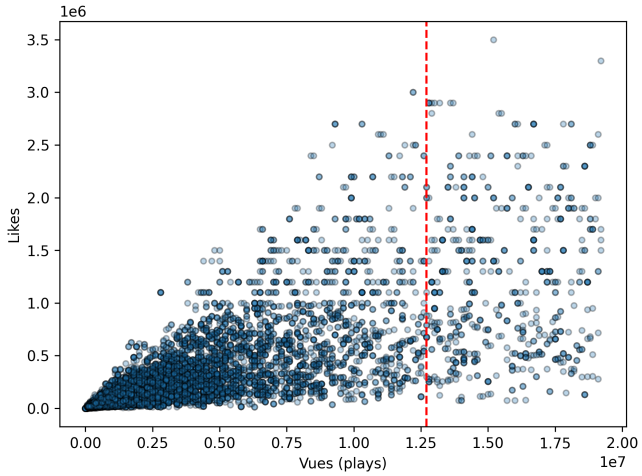

Figure 8: Dispersion *likes* / *views* (filtrée au 95^{e} percentile). La ligne verticale rouge indique le seuil 90^{e} percentile : seules les vidéos au-delà de 4 650 000 vues entrent dans la classe `is_viral`=1.

6 Analyse finale du dataset consolidé

Après les différentes étapes de préparation, nous avons procédé à une analyse approfondie du dataset consolidé afin de valider la qualité de nos variables et mieux comprendre leurs effets sur la viralité des vidéos TikTok. Nous détaillons ici les résultats clés obtenus via une approche descriptive et une régression logistique simple.

6.1 Analyse avancée de la corrélation (Phi)

Une matrice de corrélation basée sur le coefficient Phi (ϕ), particulièrement adapté aux données binaires, a été calculée afin d'identifier les redondances potentielles entre variables explicatives. Les paires de variables avec $\phi > 0.3$ sont les suivantes :

Variable 1	**Variable 2**	**Phi (ϕ)**
hr_afternoon	hr_evening	0,698
tag_foryou	tag_foryoupage	0,385
tag_fyp	tag_viral	0,328
tag_fyp	tag_foryou	0,323
tag_fyp	tag_fy	0,313
hr_morning	hr_evening	0,308
tag_viral	tag_fy	0,307
hr_morning	hr_afternoon	0,304
tag_foryou	tag_viral	0,304

Table 1: Corrélations Phi (ϕ) entre variables catégorielles binaires (seuil $\phi > 0,3$).

Cette analyse met en évidence une très forte corrélation entre certaines catégories temporelles (*après-midi*/*soir*) ainsi qu'entre hashtags populaires (`#foryou`, `#foryoupage`, `#fyp`), ce qui pourrait orienter nos choix de sélection de variables en modélisation afin d'éviter la redondance informationnelle.

6.2 Prévalence des variables selon la viralité

Nous avons ensuite comparé la proportion de `True` pour chaque variable binaire selon la classe `is_viral`. Deux exemples illustratifs sont représentés ci-dessous :

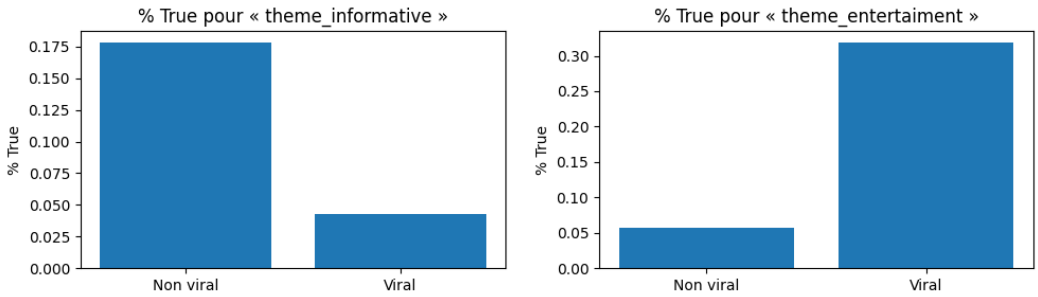

Figure 9: Prévalence comparée de deux thèmes selon viralité.

On observe notamment que :

- Le thème `entertainment` est très sur-représenté chez les vidéos virales ($> 30\,\%$ contre $\sim 5\,\%$ côté non viral).
- À l'inverse, le thème `informative` est davantage lié à des vidéos non virales.

6.3 Analyse de la durée des vidéos

La durée des vidéos semble également jouer un rôle notable dans la viralité. La Figure 10 présente la distribution des durées selon la cible `is_viral` :

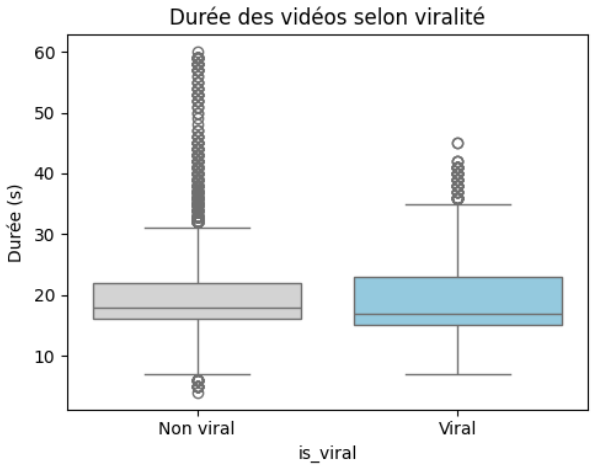

Figure 10: Comparaison des durées vidéos selon viralité (boxplot).

Les vidéos virales présentent une légère tendance à être plus longues : la médiane est légèrement supérieure et la dispersion plus importante. Cependant, de nombreuses vidéos courtes peuvent également atteindre la viralité, ce qui justifie l'inclusion de la variable `duration` dans les modèles mais sans en attendre un effet prédominant.

6.4 Importance des variables (régression logistique)

Une première modélisation via une régression logistique non régularisée nous a permis d'identifier les variables ayant le plus fort impact sur la viralité, en valeur absolue de leur coefficient.

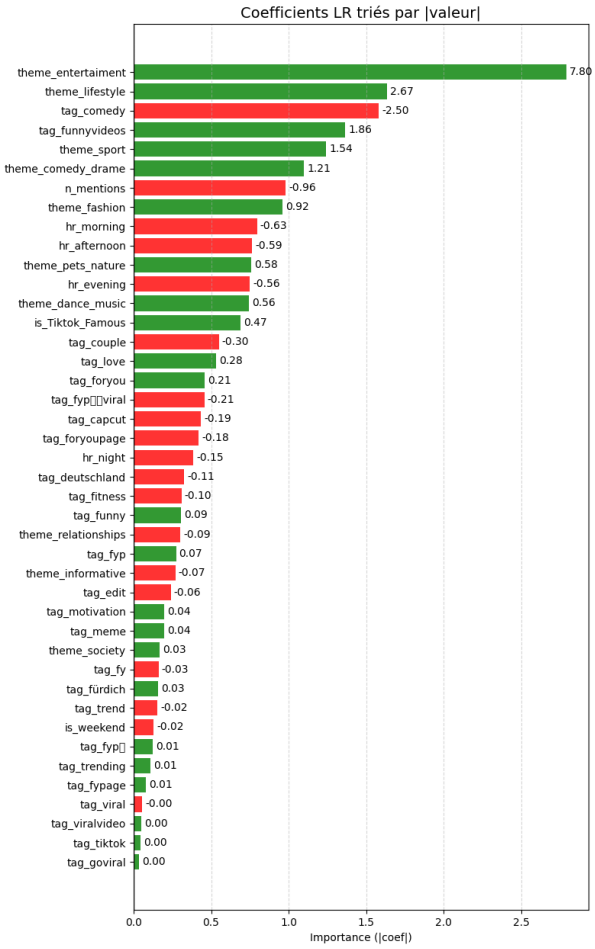

Figure 11: Top 30 des variables les plus influentes (régression logistique).

Les résultats montrent que :

- Les thèmes `entertainment`, `lifestyle` et `sport` ont un impact très positif.
- À l'inverse, les hashtags `#comedy` ou la présence de mentions (`n_mentions`) ont tendance à être corrélés négativement à la viralité.
- Le statut `is_Tiktok_Famous` reste un booster modéré de viralité, ce qui valide notre hypothèse initiale.
- Le créneau horaire de publication (`hr_morning`, `hr_evening`) joue également un rôle négatif.

Cette analyse permet de dégager plusieurs pistes de réflexion métier (quels thèmes favorisent la viralité) et oriente le choix de variables pour les modèles prédictifs supervisés ultérieurs.

float

7 Méthodologie expérimentale

7.1 Partitionnement des données

Pour évaluer la capacité de généralisation de nos modèles tout en tirant pleinement parti du jeu de données, nous utilisons :

- **Stratified Train/Test Split** : un découpage simple à 80 % pour l'entraînement et 20 % pour le test, en préservant la proportion de vidéos virales et non virales. Rapide à exécuter, il reste sensible au hasard du découpage.
- **Stratified 5-fold Cross-Validation** : le jeu est divisé en cinq sous-ensembles (plis). À chaque itération, un pli sert de validation et les quatre autres d'entraînement, avec conservation de la proportion de chaque classe.

Choix retenu : la *stratified 5-fold CV*, car elle réduit la variance d'évaluation et assure une estimation plus stable, surtout pour un jeu déséquilibré.

7.2 Modèles testés

Nous avons comparé cinq méthodes de classification simples et rapides à entraîner :

LDA (LinearDiscriminantAnalysis) *Principe* : les deux classes sont modélisées par des distributions normales multivariées avec même covariance. La frontière de décision est linéaire.

- *Avantages* : prend en compte la covariance, bon compromis biais/variance si l'hypothèse gaussienne est valide.
- *Limites* : matrice de covariance identique pour chaque classe, sensible aux outliers.

QDA (QuadraticDiscriminantAnalysis) *Principe* : comme LDA, mais chaque classe a sa propre matrice de covariance, ce qui produit une frontière quadratique.

- *Avantages* : plus flexible que LDA en cas de formes de distribution différentes.
- *Limites* : plus de paramètres à estimer, vulnérable si peu de données par classe.

Logistic Regression (`log_reg`) *Principe* : modélise le log-odds de la viralité comme combinaison linéaire des variables.

- *Avantages* : interprétation directe des coefficients, calibration des probabilités.
- *Limites* : suppose une relation linéaire dans l'espace des log-odds, sensible aux corrélations élevées.

Gaussian Naive Bayes (`GNB`) *Principe* : assume l'indépendance conditionnelle des variables et une loi normale pour chaque caractéristique dans chaque classe.

- *Avantages* : très rapide, fonctionne bien sur petits ensembles de données.
- *Limites* : hypothèse d'indépendance souvent violée, ce qui peut dégrader les performances.

K-Nearest Neighbors (`KNN`) *Principe* : classe une nouvelle vidéo selon la majorité des k plus proches voisines dans l'espace des features (ici $k = 5$).

- *Avantages* : non paramétrique, s'adapte à toute forme de frontière de décision.
- *Limites* : coût de prédiction élevé, sensible au choix de k et à l'échelle des variables.

7.3 Optimisation des hyperparamètres

Pour chaque méthode, nous exécutons une recherche systématique (`GridSearchCV` de `scikit-learn`) :

- **LDA/QDA** : test de différents `solver` (`svd`, `lsqr`, `eigen`) et de niveaux de `shrinkage` pour LDA.
- **LogisticRegression** : grille de `C` (de 10^{-4} à 10^{2}), pénalités `l1`/`l2`, choix de `solver`.
- **GaussianNB** : ajustement du `var_smoothing`.
- **KNN** : `n_neighbors` entre 3 et 15, poids `uniform` ou `distance`.

Chaque configuration est validée selon la *stratified 5-fold CV* et sélectionnée sur la base du score *AUC*.

7.4 Critères d'évaluation des modèles

Nous retenons l'**AUC (Area Under the ROC Curve)** comme métrique principale :

- *Seuil indépendant* : évalue la qualité du classement sans fixer de seuil de décision.
- *Robustesse au déséquilibre* : reste fiable lorsque les classes sont inégales.
- *Interprétation intuitive* : probabilité qu'une vidéo virale soit classée au-dessus d'une vidéo non virale.

8 Résultats expérimentaux

8.1 Performance des modèles (accuracy)

Nous avons évalué chacun des cinq classifieurs en mesurant leur *accuracy* (proportion d'échantillons correctement prédits) sur le jeu de test. Le graphique ci-dessous montre la distribution des scores obtenus sur plusieurs répliques ou plis.

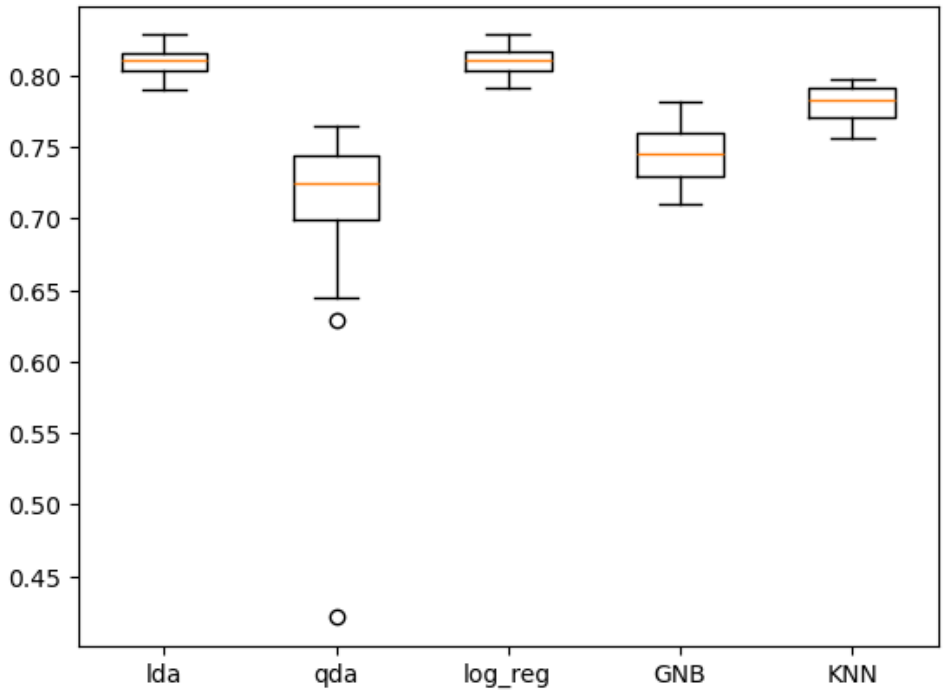

Figure 12: Distribution des scores d'accuracy en test pour chaque modèle.

On constate que la *Logistic Regression* et le *KNN* sont légèrement en tête, tandis que le *QDA* présente la plus grande variabilité de performance.

8.2 Coefficients de la régression logistique

Après avoir entraîné la régression logistique, nous avons extrait la signification des classes et les coefficients associés à chaque variable explicative :

```
Mapping des classes : {0: 'non-viral', 1: 'viral'}
```

Feature	Coef.
theme_entertaiment	2.778
theme_informative	-1.068
theme_relationships	-0.909
theme_sport	0.636
theme_lifestyle	0.526
...	...
n_hashtags	0.024
tag_trending	-0.021
hour	-0.018
tag_fypviral	0.0005

Table 2: Principaux coefficients de la régression logistique.

8.3 Interprétation métier

- Les variables de type `theme_...` ressortent en tête : certains thèmes (*entertainment*, *sport*) augmentent nettement la probabilité de viralité, tandis que d'autres (*informative*, *relationships*) la diminuent.
- Le nombre de hashtags (`n_hashtags`) a un effet positif modéré : plus il y a de hashtags, plus la portée peut s'étendre.
- L'heure de publication (`hour`) a un coefficient légèrement négatif, suggérant qu'un post tardif est un peu moins susceptible de devenir viral.
- Les tags spécifiques (`tag_trending`, `tag_fypviral`) ont un impact très faible, ce qui peut indiquer une moindre valeur ajoutée de ces seuls marqueurs.

Ces résultats confirment que, pour ce jeu de données, les variables thématiques et l'engagement initial (via les hashtags) sont des leviers clés pour prédire la viralité.

9 Modèles d'arbres et méthodes de boosting

9.1 Decision Tree (Arbre de décision)

L'arbre de décision est un modèle simple qui divise progressivement les données en groupes plus petits, en choisissant à chaque étape la question la plus utile pour séparer les vidéos virales des non-virales. Concrètement, on part de toutes les observations et l'on cherche, parmi toutes les variables et tous les seuils possibles, celui qui crée deux sous-ensembles où l'on retrouve le plus de résultats identiques (par exemple viral d'un côté, non-viral de l'autre). On répète ce processus jusqu'à ce que chaque groupe soit suffisamment pur ou qu'on atteigne une limite de profondeur.

Pour éviter que l'arbre n'apprenne le bruit ou les cas particuliers, on fixe quelques règles : on peut limiter le nombre de divisions successives (profondeur maximale), imposer un nombre minimal d'exemples pour faire une coupe, ou même supprimer après construction les branches trop fines (élagage). Ces paramètres protègent contre l'excès d'ajustement et rendent le modèle plus stable.

L'un des grands avantages de l'arbre de décision est son extraordinaire clarté : chaque chemin de la racine à une feuille se traduit en une règle du type si durée 30 s et nombre de hashtags plus grand que 5 alors viral . Cela aide les équipes métier à comprendre et à valider le modèle. De plus, la prédiction est très rapide, car il suffit de parcourir quelques questions pour arriver à une réponse.

En revanche, si on ne régularise pas assez l'arbre, il devient trop spécifique aux données d'entraînement et peut sombrer dans le sur-apprentissage. Par ailleurs, les arbres peuvent favoriser les variables qui offrent beaucoup de coupures possibles et être sensibles aux valeurs aberrantes. C'est pourquoi, en pratique, on leur préfère souvent des méthodes ensemblistes comme la forêt aléatoire ou le boosting, qui gardent la simplicité de l'arbre tout en réduisant ses faiblesses.

9.2 Random Forest (Forêt aléatoire)

La forêt aléatoire est une méthode qui construit plusieurs arbres de décision et combine leurs prédictions pour obtenir un résultat plus fiable. Concrètement, chaque arbre est entraîné sur un sous-ensemble différent des données (tirage aléatoire avec remise) et, à chaque étape de division, il ne considère qu'un échantillon de variables au hasard. Cette double randomisation (sur les données et sur les variables) permet de réduire fortement le risque que tous les arbres fassent les mêmes erreurs ou s'ajustent trop aux particularités du jeu d'entraînement.

Pour prédire la classe d'une nouvelle vidéo, on demande à chaque arbre de voter (viral ou non-viral), puis on prend la classe qui obtient le plus de voix. Cette approche améliore la stabilité et la précision par rapport à un seul arbre : même si un arbre est mal entraîné ou sensible à un bruit local, la majorité des autres compensent.

Les principaux réglages sont le nombre d'arbres (`n_estimators`), qui contrôle la robustesse de la forêt, et la taille de l'échantillon de variables testé à chaque division (`max_features`). Plus on ajoute d'arbres, plus la variance globale diminue, mais au prix d'un coût de calcul plus élevé. Limiter `max_depth` ou le nombre minimal d'échantillons par feuille aide aussi à éviter l'overfitting sur chaque arbre individuel.

L'un des atouts majeurs de la forêt aléatoire est sa capacité à gérer de nombreux types de variables sans prétraitement complexe et à donner une mesure d'importance pour chaque caractéristique, ce qui aide à comprendre quelles variables influencent le plus la viralité. En revanche, l'interprétation globale devient moins intuitive que pour un seul arbre, et la prédiction peut être plus lente si l'on utilise des centaines d'arbres.

9.3 XGBoost (Extreme Gradient Boosting)

XGBoost est une méthode qui construit plusieurs petits arbres de décision l'un après l'autre pour corriger progressivement les erreurs de prédiction. On commence par un arbre simple, on mesure l'écart entre ses prédictions et la vérité, puis on ajoute un deuxième arbre destiné à réduire ces écidus. Chaque nouvel arbre se concentre sur les cas où les prédictions précédentes se sont trompées, et la prédiction finale est la somme des contributions de tous les arbres.

Pour éviter que le modèle ne devienne trop complexe et n'apprenne le bruit du jeu de données, XGBoost applique un petit coefficient d'apprentissage (appelé learning rate) qui réduit l'impact de chaque arbre, et une régularisation qui pénalise les arbres trop

profonds ou comportant trop de feuilles. Ces deux mécanismes garantissent que l'ajout d'un nouvel arbre apporte un vrai bénéfice sans provoquer de sur-apprentissage.

En pratique, XGBoost est célèbre pour sa rapidité et son efficacité mémoire, car l'algorithme est optimisé pour tirer parti du calcul parallèle et pour compresser intelligemment les données. Cette combinaison de rapidité et de robustesse en fait un choix fréquent pour les compétitions de machine learning, où il atteint régulièrement les meilleurs scores.

Cependant, cette puissance a un coût : XGBoost dispose de nombreux réglages (nombre d'arbres, profondeur, learning rate, paramètres de régularisation) et trouver la combinaison optimale peut prendre beaucoup de temps et de ressources. C'est pourquoi il est important de démarrer avec des valeurs raisonnables et d'ajuster progressivement les paramètres en fonction des résultats obtenus.

9.4 Intégration et évaluation dans notre pipeline

Nous commençons par analyser les courbes ROC obtenues pour les trois modèles arborés : Decision Tree, Random Forest et XGBoost. La figure 13 regroupe les courbes de chacun, avec leur AUC respectif.

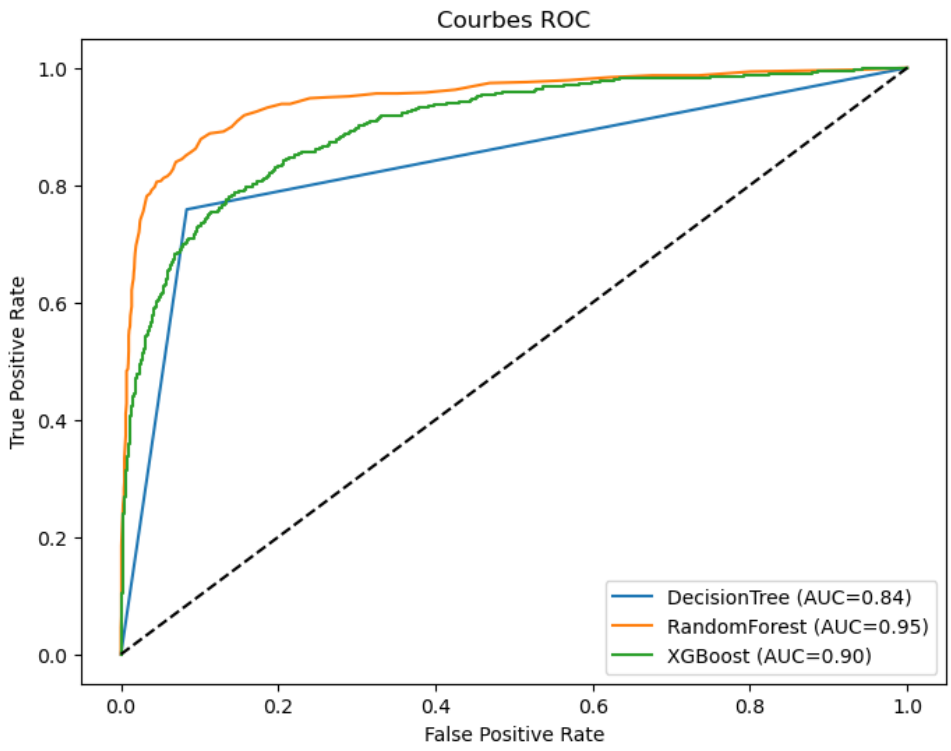

Figure 13: Courbes ROC pour Decision Tree (AUC = 0.84), Random Forest (AUC = 0.95) et XGBoost (AUC = 0.90).

Sur ce graphique, l'axe horizontal (False Positive Rate, FPR) mesure la proportion de non-viraux mal classés en viraux, et l'axe vertical (True Positive Rate, TPR) la proportion de viraux correctement identifiés. Plus une courbe est proche du coin supérieur gauche, meilleur est l'arbitrage entre sensibilité et spécificité.

Decision Tree La courbe bleue correspond à l'arbre de décision simple. On observe un AUC de 0,84, ce qui traduit une capacité raisonnable à distinguer vidéos virales et

non-virales, mais avec une pente assez linéaire : le modèle peine à atteindre un TPR élevé sans accepter un FPR important.

Random Forest La courbe orange est celle de la forêt aléatoire. Avec un AUC de 0,95, c'est clairement le meilleur modèle du lot : il atteint un TPR proche de 0,9 dès qu'on accepte un FPR inférieur à 0,2, et se stabilise très haut, signe d'une forte discrimination et d'une bonnerobustesse.

XGBoost La courbe verte illustre XGBoost, avec un AUC de 0,90. Bien qu'un peu en dessous de la forêt aléatoire, XGBoost présente un très bon compromis dès les faibles FPR (autour de 0,1) et se rapproche des performances de Random Forest sur l'ensemble de la plage.

En conclusion, la forêt aléatoire fournit la meilleure séparation globalement, XGBoost suit de près et l'arbre unique reste performant mais moins discriminant. Ces résultats confirment l'intérêt des méthodes d'ensemble pour ce jeu de données TikTok, où la diversité des arbres améliore significativement la capacité à prédire la viralité.

9.5 Analyse des matrices de confusion

Pour mieux comprendre les types d'erreurs commises par chaque modèle, nous affichons ci-dessous les matrices de confusion obtenues pour Decision Tree, Random Forest et XGBoost.

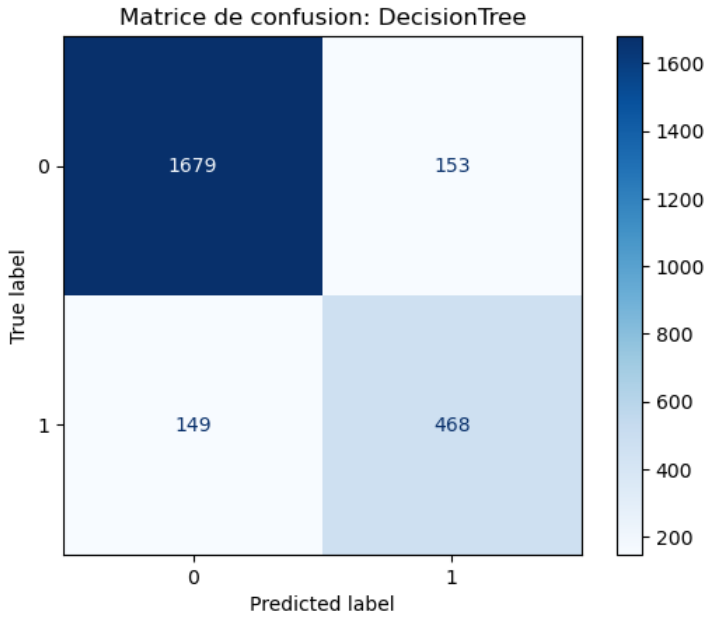

Figure 14: Matrice de confusion – Decision Tree

Sur la matrice de l'arbre de décision (figure 14), on note que sur 1832 vidéos non-virales, 1679 sont correctement classées (vraies négatives) et 153 sont faussement prédites comme virales (faux positifs). Côté viralité, 468 vidéos sont identifiées correctement (vraies positives) mais 149 virales sont manquées (faux négatifs). Cette répartition reflète l'AUC plus modérée de 0,84 : le modèle confond encore parfois les deux classes.

La forêt aléatoire (figure 15) améliore nettement la précision : seules 39 non-virales sont faussement classées virales, et 179 virales sont mal détectées, tandis que 1793 et 438

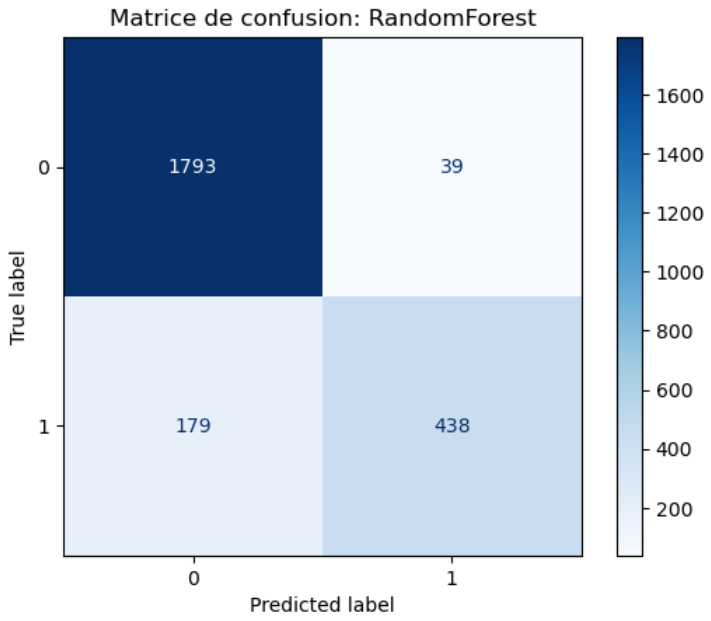

Figure 15: Matrice de confusion – Random Forest

échantillons sont parfaitement prédits. Cette forte réduction des erreurs confirme son AUC de 0,95 et sa robustesse aux faux positifs.

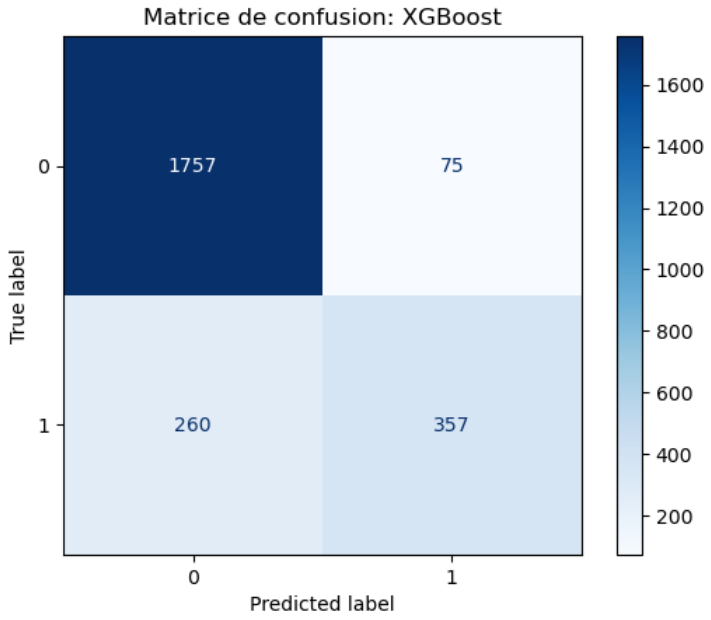

Figure 16: Matrice de confusion – XGBoost

Enfin, XGBoost (figure 16) présente un compromis : avec 75 faux positifs et 260 faux négatifs, il identifie correctement 1757 non-virales et 357 virales. Son AUC de 0,90 se traduit par un équilibre satisfaisant, même si le modèle reste un peu plus généreux en faux négatifs que la forêt aléatoire.

Dans l'ensemble, ces matrices montrent que la Random Forest est la plus fiable pour minimiser les deux types d'erreurs, tandis que XGBoost et l'arbre simple présentent

respectivement un compromis et une variabilité plus importante.

10 Support Vector Machine (SVM)

Le Support Vector Machine, ou SVM, est un algorithme de classification qui cherche à tracer, dans un espace à plusieurs dimensions, la meilleure frontière séparant deux classes. Imaginons chaque vidéo TikTok représentée par un point dans cet espace, où chaque dimension correspond à une caractéristique (nombre de likes, durée, nombre de hashtags, etc.). Le SVM détermine alors un hyperplan — c'est-à-dire une ligne ou un plan de séparation — qui maximise la distance entre lui et les points d'entraînement les plus proches de chaque classe, appelés *vecteurs de support*. Cette marge la plus large possible rend le modèle plus robuste face aux variations ou au bruit dans les données.

Quand les points ne sont pas séparables par une ligne droite, le SVM utilise le kernel trick pour projeter implicitement les données dans un espace de dimension plus élevée. Par exemple, avec un kernel gaussien (RBF), deux points sont comparés selon une mesure de proximité, ce qui permet de dessiner une frontière courbe dans l'espace d'origine sans calculer explicitement les nouvelles coordonnées. Cette astuce permet au SVM de capturer des relations non linéaires tout en restant efficace.

Dans notre projet, nous avons intégré le SVM exactement comme les autres modèles : après avoir normalisé les données, nous avons choisi un kernel gaussien (RBF) pour permettre des frontières souples entre vidéos virales et non-virales. Nous l'avons d'abord évalué par validation croisée à cinq plis, afin d'en vérifier la stabilité, puis testé sur le jeu de test pour mesurer sa performance finale.

Le paramètre clé du SVM, C, règle le compromis entre deux comportements :

- quand C est élevé, on cherche à classer correctement la plupart des exemples, quitte à adapter la frontière aux détails du jeu d'entraînement,
- quand C est faible, on tolère plus d'erreurs sur l'entraînement pour obtenir une frontière plus générale, moins sensible au bruit.

Le SVM est particulièrement adapté lorsque la séparation entre viral et non-viral n'est pas linéaire mais reste simple à décrire. Sa recherche d'une marge maximale autour de l'hyperplan le rend robuste face aux exemples atypiques. En revanche, son temps d'entraînement peut devenir important si le nombre de vidéos explose ; pour notre volume actuel de données TikTok, il offre cependant un très bon équilibre entre précision et robustesse.

11 Analyse de la matrice de confusion du SVM

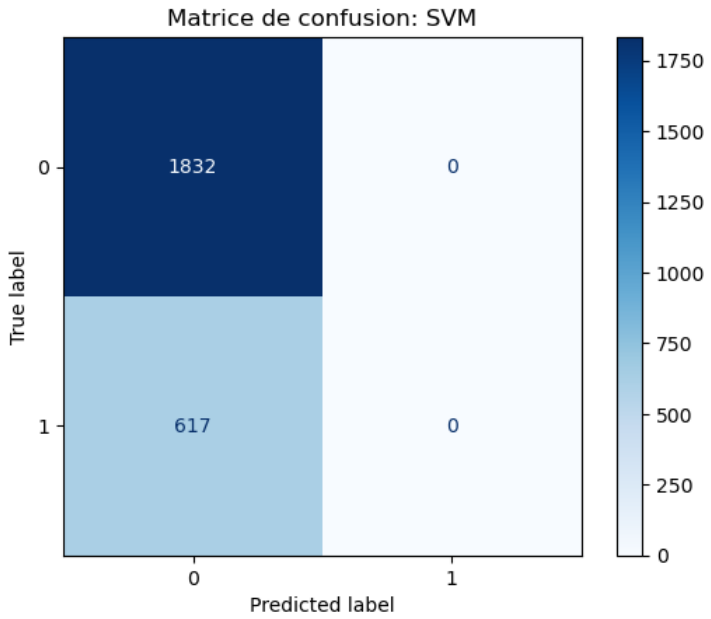

Figure 17: Matrice de confusion – SVM

Sur la figure 6, le SVM a prédit systématiquement la classe non-viral (0), ce qui se traduit par :

- 1 832 vraies négatives (vidéos non virales correctement identifiées),
- 617 faux négatifs (vidéos virales confondues avec des non-virales),
- 0 faux positifs et 0 vraies positives.

Cette configuration conduit à une *accuracy* de

$$\frac{1832}{1832 + 617} \approx 0{,}75,$$

mais le modèle n'a jamais reconnu une seule vidéo virale.

Interprétation et limites :
Le SVM, tel qu'implémenté avec un kernel RBF sans optimisation de paramètres, se comporte presque de manière aléatoire pour la classe minoritaire (virale). En présence de recherche d'hyperparamètres (GridSearchCV), le coût de calcul aurait été trop élevé, donc nous avons conservé des valeurs par défaut, ce qui explique ce déséquilibre extrême.
Pour améliorer ces résultats, il serait nécessaire de :

- lancer une recherche de grille (GridSearchCV) sur C et γ,
- ajuster la balance de classes ou appliquer un sur-échantillonnage des vidéos virales,
- envisager un kernel plus adapté ou un modèle plus flexible.

12 Gradient Boosting et Stacking

Dans cette section, nous présentons deux approches complémentaires pour améliorer la prédiction de la viralité : le *Gradient Boosting*, qui construit une suite d'arbres corrigeant à chaque fois les erreurs des précédents, et le *Stacking*, qui combine plusieurs modèles de base au sein d'un méta-modèle.

12.1 Gradient Boosting Classifier

Le Gradient Boosting est une technique de *boosting* où l'on construit successivement un ensemble d'arbres peu profonds, chacun apprenant à corriger les résidus (les erreurs) laissés par l'ensemble précédent. Dans notre implémentation, nous avons utilisé :

- 100 arbres (`n_estimators=100`),
- un taux d'apprentissage (`learning_rate=0.1`) qui réduit la contribution de chaque arbre pour stabiliser l'ensemble,
- une profondeur maximale de 3 (`max_depth=3`) pour éviter que chaque arbre ne sur-apprenne.

Concrètement, le premier arbre tente de prédire la viralité, puis on mesure la différence entre ses prédictions et la vérité. Le second arbre est entraîné sur ces erreurs, le troisième sur les erreurs résiduelles du deuxième, et ainsi de suite, jusqu'à 100. Cette approche permet d'obtenir un modèle puissant capable de capter des relations complexes tout en contrôlant le sur-apprentissage grâce au learning rate et à la faible profondeur des arbres.

12.2 Stacking Classifier

Le stacking est une technique de combinaison (*ensemble*) où l'on entraîne d'abord plusieurs modèles de base (ici un arbre de décision, une forêt aléatoire et un XGBoost), puis l'on utilise leurs prédictions comme nouvelles variables d'entrée pour un *méta-modèle*, en l'occurrence une régression logistique.

Le processus se déroule en deux temps :

1. Chaque modèle de base est entraîné en validation croisée à 5 plis sur l'ensemble d'entraînement. Pour chaque pli, on collecte les prédictions sur la partie tenue à l'écart, ce qui évite toute fuite d'information.
2. Les prédictions ainsi obtenues (trois colonnes) servent de données d'entrée pour entraîner la régression logistique finale, qui apprend à combiner les forces et à corriger les faiblesses de chaque modèle de base.

En pratique, nous avons défini la liste des estimateurs de base : `[("dt", DecisionTree), ("rf", RandomForest), ("xgb", XGBClassifier)]` et choisi `LogisticRegression(max_iter=1000, solver="liblinear")` comme méta-modèle. Ce stacking permet souvent de dépasser les performances de chacun des modèles pris isolément, car il exploite leur complémentarité.

12.3 Évaluation conjointe

Nous présentons ici les résultats globaux obtenus après avoir ajouté le Gradient Boosting, le Stacking et le SVM à notre pipeline, en comparaison avec les autres modèles.

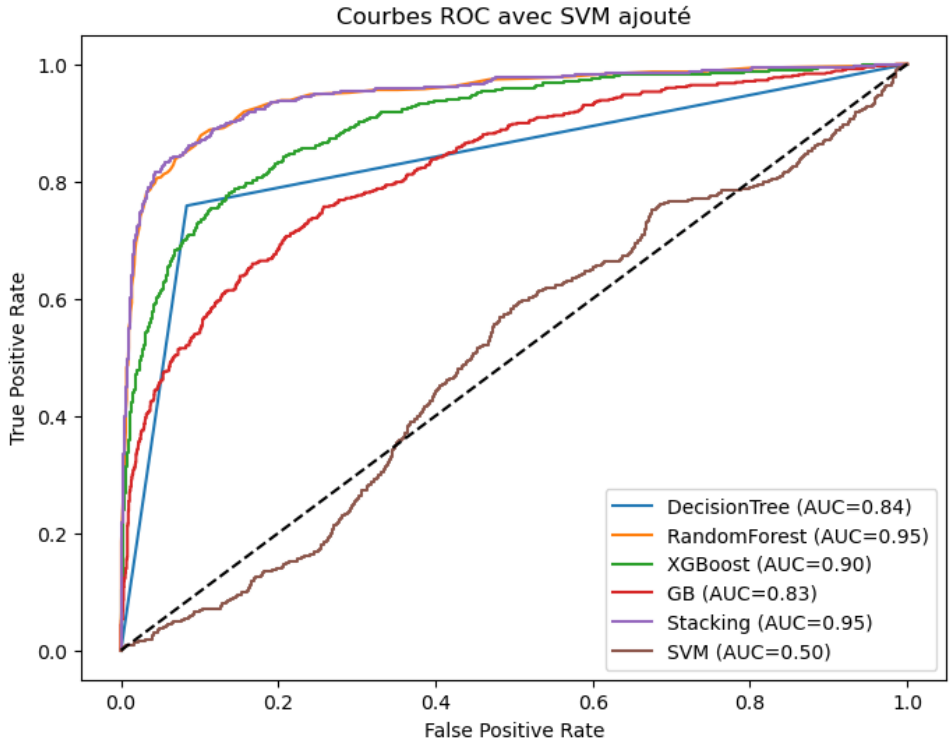

Figure 18: Courbes ROC comparant tous les modèles (DecisionTree, RandomForest, XGBoost, GB, Stacking, SVM).

La figure 18 montre que :

- **Stacking** et **RandomForest** atteignent chacune un AUC de 0,95, ce qui indique une excellente discrimination entre viraux et non-viraux.
- **XGBoost** suit avec un AUC de 0,90, confirmant sa robustesse grâce aux optimisations qu'il apporte au boosting.
- **Gradient Boosting** classique affiche un AUC de 0,83, en retrait par rapport à XGBoost en l'absence de régularisations avancées.
- **DecisionTree** obtient 0,84, montrant qu'un arbre seul reste un bon point de départ.
- **SVM** plafonne à un AUC de 0,50, révélant son incapacité à distinguer les deux classes sans réglage approfondi.

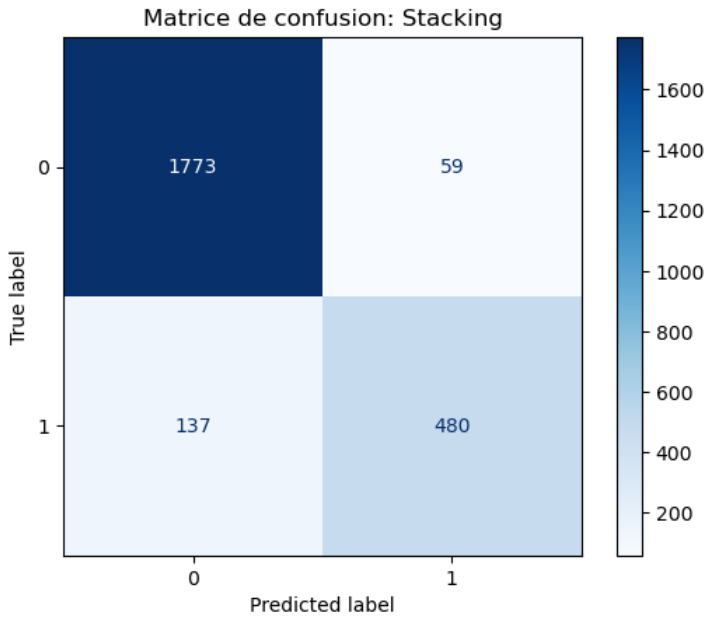

Figure 19: Matrice de confusion pour le Gradient Boosting

Matrix de confusion — Gradient Boosting Comme le montre la figure 19, le Gradient Boosting identifie correctement 1 769 vidéos non-virales et 251 vidéos virales, mais il manque 366 vidéos virales (faux négatifs) et crée 63 faux positifs. Cette répartition correspond à une bonne spécificité mais une sensibilité perfectible.

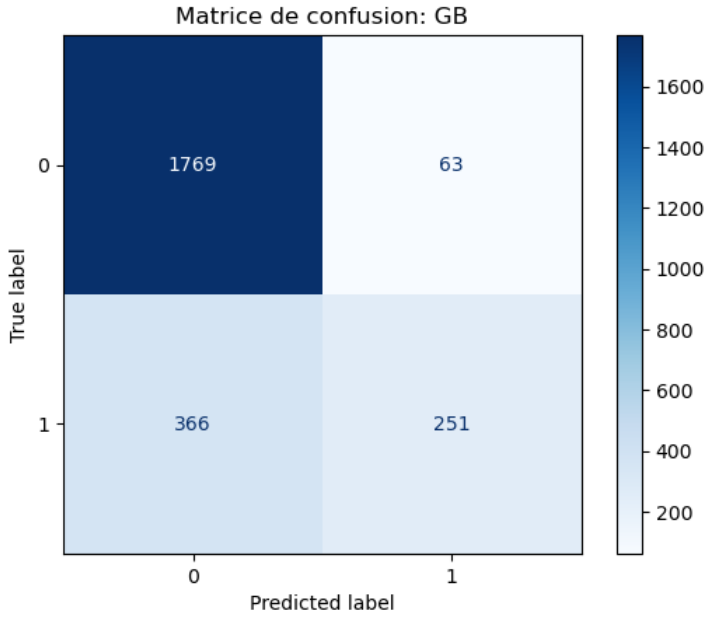

Figure 20: Matrice de confusion pour le Stacking

Matrix de confusion — Stacking La figure 20 illustre la supériorité du Stacking : avec 1 773 vraies négatives et 480 vraies positives, les faux négatifs (137) et faux positifs (59) sont très réduits. Cela confirme la forte AUC de 0,95 et un excellent compromis sensibilité/spécificité.

En résumé, l'évaluation conjointe montre que :

- Le **Stacking** tire pleinement parti de la complémentarité des modèles pour offrir la meilleure performance globale.
- Le **RandomForest** reste une valeur sûre et très compétitive.
- Le **Gradient Boosting** offre un gain notable par rapport à un arbre simple, mais reste en retrait face aux versions optimisées (XGBoost) ou combinées (Stacking).
- Le **SVM**, sans tuning précis, se comporte de manière quasi aléatoire et n'apporte pas de valeur ajoutée dans notre contexte.

13 Conclusion

Après avoir comparé un large éventail de modèles — des arbres de décision simples (LDA, QDA) et de la régression logistique jusqu'aux méthodes d'ensemble sophistiquées (Random Forest, Gradient Boosting, XGBoost, Stacking) et aux SVM — il apparaît que les modèles les plus performants sur notre jeu de données TikTok sont :

- la **Random Forest**, grâce à sa combinaison d'arbres bootstrap et à sa capacité à réduire la variance,
- le **Stacking**, qui exploite la complémentarité des arbres, de la forêt et de XGBoost pour maximiser la discrimination,
- la **LDA** (Linear Discriminant Analysis) et la **régression logistique**, qui, malgré leur simplicité, offrent un excellent compromis biais/variance lorsque la frontière entre viral et non-viral est à peu près linéaire dans l'espace transformé de nos variables.

En revanche, le **SVM** s'est révélé, dans notre implémentation sans optimisation de ses hyperparamètres, le moins performant, avec un comportement proche de la prédiction aléatoire. Ce résultat s'explique principalement par l'absence de recherche systématique de grille (GridSearchCV) sur les paramètres C et γ, dont le coût de calcul serait prohibitif à l'échelle de notre pipeline actuel. Il reste néanmoins probable qu'avec un tuning fin et adapté, le SVM puisse devenir un modèle très compétitif pour ce type de classification binaire.

En ouverture, ces conclusions invitent à deux pistes principales pour prolonger ce travail :

1. réaliser une optimisation plus poussée des hyperparamètres (notamment pour SVM et XGBoost) en utilisant des stratégies de recherche plus efficaces (Bayesian optimization, Hyperband),
2. explorer des modèles encore plus sophistiqués, tels que les réseaux neuronaux profonds ou les méthodes de meta-learning avancées, pour tenter de capturer des patterns plus subtils dans les données TikTok.