**Projet d’analyse de sentiments**

L’objectif de notre projet est de mener une analyse de sentiments sur des tweets afin d’identifier les moments de la journée où les utilisateurs sont les plus enclins à ressentir des émotions positives. Cette étude permettra ainsi de déterminer les créneaux horaires les plus propices à la diffusion de spots publicitaires, lorsque le public est le plus réceptif et susceptible d’apprécier les annonces.

Dans ce projet, nous allons donc concevoir un modèle de prédiction de sentiment, en fonction du pays et du moment de publication du tweet.

Au niveau des technologies utilisées, pour la prédiction, nous allons travailler

**1 – Interprétation des données.**

Nous avons commencé notre projet par découvrir les données de notre dataset. Ce dernier est un CSV contenant des tweets, avec leur pays de publication et le moment de la journée où ils ont été publiés.

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

Les données étaient assez claires et explicites, et nous avons donc directement voulu commencer notre analyse.

**2 – Préparation des données.**

Notre projet à été réalisé sur VSCode, et nous avons utilisé un Jupyter Notebook pour réaliser notre travail de prédiction des sentiments.

Pour commencer, nous avons donc importé notre CSV et l’avons mis dans un dataframe, dont nous avons ensuite extrait les informations (colonnes, types)

Une image contenant texte, capture d’écran, menu, Police

Le contenu généré par l’IA peut être incorrect.

Ensuite, nous avons commencé à nettoyer le dataframe fraichement créé. Pour cela, nous avons décidé d’appliquer plusieurs méthodes :

* Mapping des émojis ‘classiques’ :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

* Réduction primaire des abréviations :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

* Création de notre fonction de nettoyage comprenant :
  + La normalisation du code en ‘NFC’
  + Retirer les mentions des utilisateurs (les ‘@’)
  + Suppression des URLs
  + A l’aide de la bibliothèque ‘emoji’, nous avons transformé les émojis en caractères
  + Remplacement des émoticônes en caractères grâce à notre bibliothèque
  + Suppression des # tout en en conservant le contenu
  + Réduire les répétions de caractères
  + Tout mettre en minuscule pour harmoniser
  + Remplacement des abréviations grâce à notre bibliothèque
  + Suppression des ponctuations (sauf ‘ !’ et ‘ ?’)
  + Suppression des caractères non alphabétiques

Voici donc le résultat de notre fonction, sur un test que nous avons pu réaliser :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Et sur notre dataframe :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

**3 – Création de notre modèle de prédiction.**

Nous avons donc maintenant notre nouvelle colonne contenant notre texte nettoyé. Cependant, on a pu constater que certains tweets sont vides, ce qui peut nous poser problème par la suite. Nous avons donc décidé de retirer ces éléments nulls.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Maintenant que les tweets de notre dataframe sont prêts, nous pouvons nous attaquer à la vérification de la colonne sentiment, essentiel à notre entrainement de modèle.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

On a pu constater qu’il y a quelques valeurs en trop, que nous avons directement filtrées :

Une image contenant texte, capture d’écran, Police, affichage

Le contenu généré par l’IA peut être incorrect.

Maintenant, nous pouvons commencer à entrainer notre modèle pour faire un prédiction sur les sentiments des tweets :

**Étapes du Pipeline d’analyse :**

1. Préparation des données

La première étape consiste à nettoyer et organiser les données brutes en utilisant notre fonction de nettoyage :

* Suppression des lignes incomplètes ou présentant des valeurs manquantes pour les colonnes principales (texte du tweet et sentiment associé).
* Analyse de la distribution des différentes classes de sentiments pour s’assurer d’un équilibre (ou au moins de la prise en compte d’un éventuel déséquilibre).
* Séparation des variables indépendantes (X : textes) et dépendantes (y : sentiments).

2. Division en ensembles d’apprentissage et de test

Afin d’évaluer objectivement les performances des modèles, les données sont séparées en :

* Ensemble d’apprentissage (80 % des données)
* Ensemble de test (20 % des données)  
  La stratification est utilisée pour conserver la même proportion de chaque classe de sentiment dans les deux ensembles.

3. Vectorisation des textes

Comme les algorithmes de machine learning ne comprennent que des nombres, les textes sont transformés en vecteurs numériques à l’aide de la méthode TF-IDF (Term Frequency – Inverse Document Frequency). Cette méthode permet :

* De ne retenir que les mots et groupes de mots (bigrammes) les plus significatifs,
* D’ignorer les mots trop rares ou trop fréquents,
* De supprimer les mots vides (stopwords) de la langue anglaise.

4. Entraînement de plusieurs modèles

Quatre modèles de classification supervisée sont entraînés :

* Naive Bayes
* Régression logistique
* Support Vector Machine (SVM)
* Random Forest

Chaque modèle est testé afin de comparer leurs performances sur les mêmes vecteurs TF-IDF issus de l’ensemble d’apprentissage.

5. Évaluation des modèles

Chaque modèle est évalué sur l’ensemble de test selon plusieurs métriques :

* Précision globale (accuracy)
* Rapport de classification détaillé (précision, rappel, F1-score)  
  Le modèle présentant la meilleure performance (selon la précision) est retenu pour la suite du pipeline.

6. Prédiction et utilisation du modèle

Le modèle sélectionné permet ensuite :

* De prédire le sentiment d’un nouveau texte,
* De fournir un indice de confiance (probabilité) pour chaque classe de sentiment,
* D’être sauvegardé et rechargé ultérieurement sans nécessiter un nouvel entraînement (sérialisation via pickle).

Atouts de ce pipeline

* Transparence : Toutes les étapes sont affichées et expliquées pour faciliter la compréhension et l’audit du process.
* Comparaison automatique de plusieurs modèles pour garantir le meilleur résultat, et par la suite décider vers quel modèle se tourner.
* Praticité : Prédiction simple sur de nouveaux textes avec une mesure de la confiance du modèle.

Grâce à notre première analyse, nous avons pu déterminer quel était le meilleur modèle :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Nous allons donc utiliser ce modèle et l’améliorer pour réaliser notre prédiction.

Étapes de l’Analyse

1. Extraction et classement des mots les plus importants

* Récupération des importances attribuées à chaque mot ou n-gramme par le modèle Random Forest.
* Affichage des mots ou expressions les plus décisives, triées par importance décroissante.
* Présentation sous forme de tableau, où chaque mot est accompagné d’un indicateur visuel de son poids dans la décision du modèle.

2. Statistiques descriptives sur les features

* Calcul et affichage de statistiques globales sur toutes les variables textuelles utilisées :
  + Nombre total de features,
  + Importance moyenne, médiane, maximum, minimum,
  + Pourcentage de mots ayant une importance supérieure à la moyenne.

3. Analyse détaillée par sentiment

* Pour les mots les plus importants, on teste leur influence sur la prédiction finale du modèle.
* Pour chaque classe de sentiment (positif, négatif, neutre), les mots qui maximisent la probabilité de prédiction sont identifiés.
* Cela permet de mieux comprendre quels mots orientent le modèle vers une prédiction “positive”, “négative” ou “neutre”.

4. Visualisations graphiques

* Génération d’un graphique “bar plot” : visualisation des mots les plus importants sous forme de barres classées par poids.
* Création d’un nuage de mots (wordcloud) : les mots les plus décisifs sont affichés en proportion de leur importance, facilitant une lecture rapide des tendances globales du modèle.

5. Analyse des n-grammes

* Distinction entre les types de n-grammes : unigrams (mots seuls), bigrams (groupes de deux mots), trigrams (groupes de trois mots).
* Identification des n-grammes les plus impactants pour l’analyse de sentiments.

6. Comparaison de mots opposés

* Pour mieux interpréter le modèle, comparaison de l’influence de paires de mots opposés (“love/hate”, “good/bad”, etc.).
* Évaluation directe de la différence d’impact sur la prédiction finale selon le mot utilisé dans une phrase.

Voici donc les résultats que nous avons eu :   
Une image contenant texte, menu, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

*Exemple de mots sur le sentiment positifs détectés*

Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.

*Exemple de mots sur le sentiment négatif détectés*

Maintenant qu’on a pu voir que notre modèle fonctionne très bien, nous allons voulu voir s’il s’était améliorer au niveau des statistiques globales.

Statistiques sur notre modèle :

1. Prédiction sur l’ensemble du CSV

* Tous les textes du jeu de données sont transformés en vecteurs numériques grâce au même vectoriseur TF-IDF utilisé pendant l’entraînement.
* Le modèle prédit le sentiment de chaque texte, ce qui permet d’ajouter une nouvelle colonne (“sentiment prédit”) dans le tableau des données.

2. Évaluation quantitative des performances

* Les prédictions du modèle sont comparées aux vraies étiquettes (“vérité terrain”).
* Un rapport de classification est généré, affichant :
  + La précision,
  + Le rappel,
  + Le F1-score,
  + Le support (nombre d’occurrences par classe).
* Cela permet de mesurer la performance du modèle pour chaque classe de sentiment (positif, négatif, neutre).

Pour notre test final sur l’ensemble des données, nous avons obtenu ce score :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Nous pouvons voir que notre modèle à une précision de 93%, ce qui est très bon. Le score F1 nous montre également que notre modèle est très bon sur les 3 classes.

Une image contenant texte, capture d’écran, nombre, logiciel

Le contenu généré par l’IA peut être incorrect.

Notre matrice de confusion nous confirme ces très bon résultats. Une écrasante majorité de valeurs prédites, ainsi que très peu de valeurs ‘fausses’.

Nous allons maintenant passer à l’ajout des variables de lieu et temporelles :

Pour cela, nous avons pu voir qu’il existe deux colonnes dans notre jeu de donnée, une qui indique le pays du tweet et une autre le moment de la journée.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Nous avons ici exploré notre colonne temporelle, à savoir ‘Time of tweet’. Nous pouvons voir qu’elle était déjà propre.

Nous avons ensuite fait une très courte analyse pour avoir une idée de la répartition des sentiments par moment de la journée, et nous avons pu voir que c’était assez bien réparti, avec un tendance haute pour le sentiment neutre.

Nous avons donc ensuite voulu appliquer ces variables à notre prédiction.

Pour cela, nous avons souhaité mettre notre prédiction dans un pipeline.

Une image contenant texte, capture d’écran, affichage

Le contenu généré par l’IA peut être incorrect.

Notre pipeline combine donc plusieurs données : le tweet, le pays et le moment de la journée.

Le texte est encore transformé par TF-IDF, et les variables catégorielles par OneHotEncoder.

1. Séparation des variables

Les variables explicatives utilisées sont :

* Le texte nettoyé du tweet (clean\_text\_advanced)
* Le pays d’origine (Country)
* Le moment de la journée (Time of Tweet)

La variable à prédire est le sentiment du tweet (sentiment).

2. Prétraitement automatisé

Nous utilisons un ColumnTransformer afin d’appliquer des transformations différentes selon le type de variable :

* TfidfVectorizer pour le texte : transforme chaque tweet en un vecteur numérique de poids de mots (méthode TF-IDF), permettant d’analyser la structure linguistique du message.
* OneHotEncoder pour les variables catégorielles (pays et moment) : chaque modalité (par exemple chaque pays, ou chaque créneau horaire) est transformée en une colonne binaire, indiquant la présence ou l’absence de cette modalité pour chaque observation.

3. Pipeline global

L’ensemble des opérations est enchaîné dans un Pipeline scikit-learn qui comprend :

* Le prétraitement des données via le ColumnTransformer
* L’entraînement d’un RandomForestClassifier, un modèle d’arbres de décision particulièrement adapté pour gérer des données mixtes (textes et catégories)

Ce pipeline permet de garantir l’ordre et la cohérence des traitements, aussi bien lors de l’entraînement que lors de l’application du modèle à de nouvelles données.

4. Entraînement

L’appel à la méthode fit() du pipeline permet d’appliquer automatiquement tous les prétraitements, puis d’entraîner le modèle de classification sur les données transformées.

Maintenant que notre pipeline est bien enregistré, nous pouvons concevoir notre API pour qu’elle utilise notre modèle pour pouvoir répondre aux questions.

**4 – Mise en place de l’API.**

Pour notre API, le but était de pouvoir prédire un sentiment en fonction d’une phrase entrée par l’utilisateur. Nous avons ainsi créé plusieurs routes pour répondre au besoin :

* D’obtenir automatiquement le sentiment prédit d’un tweet ou d’une phrase via notre modèle de machine learning entraîné,
* D’offrir un mode conversationnel (“chatbot”) guidant l’utilisateur étape par étape dans la formulation de son tweet pour atteindre un objectif émotionnel spécifique.

Fonctionnement global

1. Chargement du modèle de machine learning

Au démarrage, le pipeline de machine learning (sauvegardé avec joblib) est chargé en mémoire. Cette étape unique garantit la rapidité des prédictions, puisque le modèle n’a pas besoin d’être réentraîné à chaque requête.

2. Définition des données attendues

Grâce à la librairie Pydantic, l’API valide et structure automatiquement les données reçues (texte du tweet, pays d’origine, moment de la journée).

3. Les routes principales de l’API

a) Prédiction directe (/predict)

* Méthode : POST
* Entrée : Un texte, le pays et le moment de la journée sous forme de données JSON.
* Traitement :
  + Construction d’un DataFrame à partir des données reçues.
  + Application du modèle de machine learning pour obtenir le sentiment prédit et les probabilités associées à chaque classe.
  + Enregistrement facultatif de la requête dans un fichier CSV pour analyse ultérieure.
* Sortie : Un objet JSON contenant le sentiment prédit et les probabilités associées.

b) Chatbot conversationnel (/chatbot)

* Méthode : POST
* Fonctionnement :
  + Dialogue interactif, étape par étape, avec l’utilisateur (texte, pays, moment, objectif émotionnel).
  + À chaque étape, l’API valide la réponse et relance si besoin.
  + À la dernière étape, le modèle analyse le tweet proposé et conseille l’utilisateur sur la meilleure manière d’atteindre son objectif de sentiment (par exemple : reformuler le texte pour obtenir une tonalité “positive”).
  + Possibilité de recommencer le processus à tout moment via la commande “restart”.

Pour finir, pour la partie front end, nous avons décidé de faire une application Streamlit.

Nous avons donc réalisé différentes pages, avec des informations différentes :

* Une page de prédiction de sentiment simple
* Une page d’analyse des données
* Une page de données brutes
* Une page d’interaction avec notre chatbot

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Notre page de prédiction :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Notre page d’analyse de données :

Cette dernière va récupérer les informations de notre CSV principal.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Notre page de données brutes :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Notre page de chatbot :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Exemple de conversation :   
  
Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.