Développer une application Web basée sur l'API Apache KAFKA Stream, permettant de faire l'analyse des données, dans le but de **« prédire en temps réel le désabonnement des clients d'une entreprise »**

Exemple du probleme :

Cette agence de marketing travaille avec plusieurs clients. Ces clients font appel aux services de l'agence de marketing pour créer des publicités, et ces publicités sont destinées aux sites web des clients eux-mêmes. Ils ont remarqué qu'ils perdent beaucoup de clients, c'est-à-dire que de nombreux clients arrêtent d'acheter leurs services. Actuellement, ils assignent simplement **des gestionnaires de compte** de manière aléatoire. Cependant, ils ont besoin d'**un modèle d'apprentissage automatique qui les aidera à prédire quels clients risquent d'abandonner** (d'arrêter d'acheter leurs services). **Cela leur permettrait d'assigner correctement les clients les plus à risque à un gestionnaire de compte spécifique, afin de tenter de les retenir**.

gestionnaire de compte : sont des professionnels chargés de gérer la relation entre l'entreprise (dans ce cas, l'agence de marketing) et ses clients. Leur rôle principal est de servir d'intermédiaires entre l'entreprise et le client, en s'assurant que les besoins du client sont compris et satisfaits,Ils entretiennent des relations avec les clients, communiquent régulièrement avec eux, et veillent à ce que les clients soient satisfaits des services fournis. Ils travaillent à comprendre les besoins spécifiques de chaque client et s'assurent que les services offerts répondent à ces besoins.

Name (Nom) : Il s'agit de la personne qui représente ou travaille pour l'entreprise cliente qui utilise le service ou qui a été impliquée
Age (Âge) : L'âge du client.
Total_Purchase (Achat Total) : Le nombre total d'annonces publicitaires achetées par le client.
Account_Manager (Gestionnaire de compte) : Champ binaire où 0 représente l'absence de gestionnaire de compte et 1 signifie qu'un gestionnaire de compte lui a été assigné.
Years (Années) : Nombre total d'années pendant lesquelles le client a été fidèle à l'entreprise.
Num_sites (Nombre de sites) : Nombre de sites utilisant le service de l'entreprise cliente.
Onboard_date (Date d'intégration) : La date à laquelle le nom du dernier contact a été intégré ou enregistré.
Location (Emplacement) : Adresse du siège social du client.
Company (Entreprise) : Nom de l'entreprise cliente.
Churn (Résiliation) : Valeur binaire représentant si le client a résilié (1) ou non (0).

« Les données sont un atout très précieux qui offre un avantage concurrentiel dans le monde des affaires. »

Apache Kafka est une plateforme de streaming distribuée, conçue pour gérer le traitement et la transmission de flux de données en temps réel. Voici quelques cas d'utilisation courants de Kafka :

1. **Messagerie en temps réel :** Kafka est utilisé comme un système de messagerie pour la transmission de messages en temps réel entre applications. Il permet de passer des messages de manière asynchrone tout en garantissant la fiabilité et la scalabilité.
2. **Streaming de données :** Kafka est utilisé pour la transmission et la gestion de flux de données en temps réel. Les entreprises peuvent collecter, traiter et distribuer des données en continu pour divers cas d'usage, tels que l'analyse en temps réel, la surveillance, les tableaux de bord, etc.
3. **Analyse en temps réel :** Les données collectées peuvent être immédiatement analysées à mesure qu'elles arrivent dans Kafka. Cela permet des analyses en temps réel pour prendre des décisions immédiates, comme la détection d'anomalies, la personnalisation en temps réel, etc.
4. **Journalisation des événements (Loggins) :** Kafka est souvent utilisé pour journaliser des événements et des messages. Il offre une solution robuste pour conserver les journaux de toutes les activités d'un système, ce qui est utile pour le débogage, l'audit, la conformité et la relecture des données.

Dans Apache Kafka, un "topic" est une catégorie de flux de messages. C'est une entité à laquelle les producteurs envoient des données et à partir de laquelle les consommateurs récupèrent des données. Les messages envoyés à un topic sont conservés de manière persistante, et les consommateurs peuvent lire les messages à partir du début ou bien à un moment spécifique, en fonction de leur position dans le flux. Les producteurs de messages envoient des messages à un topic spécifique, et les consommateurs intéressés par un sujet particulier se connectent au topic correspondant pour lire les messages qui y sont stockés. Cela permet une séparation logique des flux de messages, ce qui facilite la gestion et le traitement des données en fonction de leurs sujets respectifs.

1. **Partitions :** Chaque topic est divisé en une ou plusieurs partitions. Les partitions sont les unités de stockage des messages dans Kafka. Elles permettent de répartir les données, d'assurer l'évolutivité et la tolérance aux pannes. Chaque partition est une séquence ordonnée et immuable de messages. Les messages au sein d'une partition sont indexés par un offset, qui représente la position d'un message spécifique dans cette partition.
2. **Broker :** Un broker est un serveur Kafka qui stocke les données. Il est responsable de la gestion des partitions des topics, du stockage et de la distribution des messages. Plusieurs brokers forment un cluster Kafka. Chaque broker peut gérer un ensemble de partitions de différents topics. Les brokers permettent de distribuer les données, de répliquer les partitions pour la redondance et de maintenir la disponibilité du système.

La relation entre ces trois concepts est la suivante :

- Les topics sont constitués de une ou plusieurs partitions.
- Chaque partition est gérée et stockée par les brokers du cluster Kafka.
- Les brokers sont responsables de la gestion des partitions et du stockage des messages pour ces partitions.

1. Création d'un topic :

Créer un topic nommé "topic_exemple" avec 3 partitions et un facteur de réplication de 2 (deux copies de chaque partition)

kafka-topics.sh --create --topic topic_exemple --partitions 3 --replication-factor 2 --bootstrap-server localhost:9092

Cette commande crée un topic appelé "topic_exemple" avec trois partitions et un facteur de réplication de 2. Cela signifie que chaque partition aura deux répliques pour la tolérance aux pannes.

2. Description des partitions du topic :

Décrire les détails des partitions pour le topic "topic_exemple"

kafka-topics.sh --describe --topic topic_exemple --bootstrap-server localhost:9092

Cette commande permet de voir les détails des partitions du topic "topic_exemple", y compris les informations sur les répliques, les leaders, etc,Cette commande indiquera les détails de chaque partition, y compris quel broker agit en tant que leader pour chaque partition et où se trouvent les répliques pour chaque partition. Les brokers spécifiques responsables de la gestion des partitions seront mentionnés dans les résultats de cette commande.

Dans Kafka, la répartition des partitions se fait de manière équilibrée entre les brokers disponibles dans le cluster.(Pour illustrer cela, si vous avez créé un topic avec 3 partitions et un facteur de réplication de 2 , chaque partition sera dirigée par un broker leader et aura une réplique sur un autre broker.)

https://hmartins-prado.medium.com/capstone-project-to-predicting-churning-customers-inside-21st-century-b25b30829811

https://github.com/Nassima-el-jazouli/Consult-Patient-Backend/blob/master/kafka/kafka-producer.py

https://chat.openai.com/share/b0544a9d-6a6f-42d4-a181-2c53d4000d0c

https://chat.openai.com/share/d299c004-581c-411a-a3b3-01cea6677da2

- https://frankdatascience.wordpress.com/2020/11/05/pyspark-modelling-for-big-data-with-databricks-and-aws-s3/