Loïse ZHU (Master SAR)
Lenaïg TASSO (Master DAC)

Projet 2 BDLE : réalisation d'un pipeline ML pour entraîner un arbre de décision pour la régression à l'aide de l'API Spark ML

lien vers le notebook :
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3505958717772969/3134175105689354/1004981343276610/latest.html

1. **Source : lien vers la source publique des données ou lien de téléchargement si pas de lien publique**

https://www.kaggle.com/jessemostipak/hotel-booking-demand

2. **Taille : en MB et en nb_lignes**

La taille du dataset est de 16.07 MB et 119390 lignes (avec 32 colonnes).

3. **Description brève des données, quelle est la variable à prédire**

Il s'agit d'un dataset sur des réservations d'hôtel dans plusieurs pays du monde. Le dataset nous donne différentes réservations sur le type d'hôtel, le type de voyageurs (combien d'adultes, d'enfants, si ce sont des visiteurs réguliers, quelle est la raison de leur voyage), ou encore le détail de la réservations (nombre de jours de réserver (weekend ou semaine), statut de la réservation, si il y a un repas ou une place de parking comprise…).
Ici, on va prédire le nombre de jours réservés pour une réservation en fonction de différents paramètres. Il y a une colonne pour le nombre de jours de la semaine et une colonne pour le nombre de jours du weekend, ainsi pour avoir le nombre total de jour, on va additionner ces deux colonnes pour créer notre nouvelle colonne "totalday".

4. **L'argumentaire demandé dans chacune des tâches décrites ci-dessous**

Tâche 1 :

Ici, on va sélectionner un sous-ensemble d'attributs qui va nous sembler pertinent pour savoir combien de jours est resté une personne ou un groupe de personnes.

- country : pertinent car selon les pays, le nombre de jours de vacances peut être différents et donc les voyages (s' ils sont de loisirs) n'auront pas la même durée.
- lead time : le temps entre la réservation et la date d'arrivée du ou des voyageurs. Plus on réserve à l'avance, plus on veut avoir de chances d'avoir sa plage de jours libres et plus on veut que sa plage de jours soit libre, plus le nombre de jours est important.
- group-type : il s'agit ici d'une colonne crée par nos soins (en fonction de si il s'agit d'une personne seule ("solo"), s'il il y a deux adultes ("couple"), ou si il y a au moins un adulte et un enfant ou un bébé ("famille"), s'il y a plusieurs adultes ("group") et pour les cas bizarres comme avoir un enfant ou bébé seul sans adults ("autres").
- arrival_date_month : le mois d'arrivé pour le séjour réservé. On peut se dire que s'il s'agit d'un séjour pendant des vacances d'été, la durée de la réservation peut être plus grande (attention, la période de l'été est différente, elle n'est pas commune pour tout le dataset car nous avons des pays des deux hémisphères.
- hotel : quel est le type de l'hôtel, la durée du séjour peut varier en fonction du type, par exemple, on peut rester plus longtemps dans un hotel moins cher mais moins longtemps dans un hôtel de basse qualité.
- is_repeated_guest : cet attribut peut aussi jouer sur notre modèle car si un visiteur est régulier, alors il est surement satisfait du service qu'il a pu tester précédemment ou par exemple que ça peut être quelqu'un qui rend visite à une connaissance pour un week-end… etc
- deposit-type : si il y a une garantie déclarée, alors on peut se dire que plus le séjour est cher, plus le séjour est long et plus on a envie de se protéger d'une annulation de dernière minute qui ne sera pas remboursée.
- agent : si la réservation a eu lieu à travers une agence, plus le séjour est long, plus on veut qu'il soit bien gérer et donc qu'il soit organisé par une agence.

Tâche 2 :
Nos features catégorielles sont : country, group_type, customer_type, arrival_date_month, hotel, is_repeated_guest, deposit_type, agent.
Notre feature continue est le lead_time.

Pour notre modèle nous avons pour :

	RMSE	MAE
train	**2.1677**	**1.4258**
test	**2.1748**	**1.4400**

Notre modèle n'est pas très précis étant donné qu'il nous donne une erreur un peu près équivalente à la variance pour le RMSE.
Cependant, avec la métrique MAE on a un modèle de meilleure qualité.

Tâche 3a :

Pour notre meilleur modèle nous avons :

	RMSE	MAE
train	**2.1770**	**1.4296**
test	**2.1588**	**1.4399**

Ces erreurs sont légèrement en dessous de celles obtenues auparavant lors de la tâche 2.

Tâche 3b :

Pour améliorer la précision nous avons décidé de prendre un sous-ensemble de données avec les réservations qui n'ont pas été annulées. En effet, ce sous-ensemble de données va permettre d'échapper aux erreurs lors des calculs des réservations sur la date d'arrivée et la date de départ ou encore le nombre de voyageurs… Ainsi on aura dans notre sous-ensemble seulement les réservations avec des voyages qui ont vraiment eu lieu. Cela nous empêchera de même d'avoir plusieurs lignes sur des réservations concernant le même voyage avec des réservations qui ont été annulées. Cela

permettra de supprimer la plupart des valeurs aberrantes.

On va de plus imputer nos valeurs manquantes. Dans notre dataset, il y a des valeurs nulles seulement au niveau des features catégorielles. Ainsi, il ne nous est pas demandé d'utiliser la classe Imputer de Spark. En effet, celle-ci permet d'imputer des valeurs des colonnes de type numérique.
La seule colonne de notre sous-ensemble contenant des valeurs nulles est la colonne "agent". Pour imputer les valeurs nulles de cette colonne, on va utiliser un algorithme stochastique.

Pour notre modèle avec les données nettoyées selon le protocole indiqué précédemment nous avons :

	RMSE	MAE
train	2.0955	1.4075
test	2.2109	1.4435

Avec notre dataset nettoyé on a une erreur qui baisse en train avec la métrique RMSE mais qui augmente lors du test. Cela est sûrement dû au fait que lors de la sélection de notre sous-ensemble, on n'a pas pris en compte une grande partie des données et donc notre base d'entraînement à grandement diminué, ce qui peut expliquer la différence.

Tâche 4 :

La librairie ML de Spark est assez intéressante mais elle manque d'une documentation qui mériterait d'être plus approfondie avec plus d'exemples sur les utilisations de certaines classes.
Cela serait aussi pratique d'avoir des librairies de visualisation plus conséquentes, cela peut s'avérer utile pour l'analyse préliminaire des données.
Ceci-dit, une fois prise en main, on arrive à comprendre assez facilement comment la librairie fonctionne et à l'utiliser assez rapidement sans trop de soucis par rapport à d'autres librairies tel que Scikit-learn.
La librairie Spark ML possède d'ailleurs une bonne représentation des algorithmes les plus communs en comparant avec Scikit-learn.

Cependant, Spark ML manque peut être une finesse sur certains algorithmes que Scikit-Learn possède.