

# Rapport du projet de microsoft Azure ML

Master 1 MIASHS parcours MQME

Keroudine BELLADJO

7 mai 2023

# I - prédire - compte tenu des différentes variables online et offline - si la Customer Journey se terminera par une conversion ou non

1. Créer une expérience dans Microsoft Azure ML qui répondra à la 1<sup>ère</sup> problématique. L'expérience contiendra au minimum 2 algorithmes qui seront comparés.

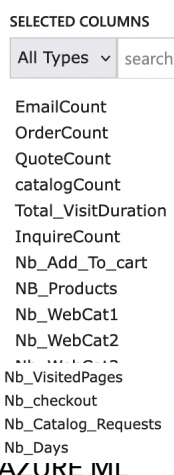
## A. Nettoyage de notre jeu de donnée

Nous avons tout d'abord chargé notre jeu de données BRADY CUSTOMER JOURNEYS puis nous avons créé une expérience que nous avons nommé « [Projet Brady](#) ». Dans un second temps nous avons procédé à l'analyse et traitement de notre jeu de données en se servant de plusieurs composants de Microsoft Azure ML:

### a. Clean Missing Data

Ce composant d'Azure ML permet de définir une opération de nettoyage. En d'autres termes il va nous permettre de supprimer, remplacer ou déduire les valeurs manquantes de notre base de données. L'objectif de ces opérations de nettoyage est d'éviter les problèmes causés par les données manquantes qui peuvent survenir lors de l'apprentissage d'un modèle. Chaque fois qu'on applique le composant [Clean Missing Data](#) à un ensemble de données, la même opération de nettoyage est appliquée à toutes les colonnes qu'on sélectionne. Par conséquent, On a donc réparti nos données en 3 groupes :

- Variables quantitatives (Les variables de comptage): Ces variables permettent pour ce jeu de données de donner la quantité ou le nombre de quelque chose. Il paraît évident que si ce nombre est manquant, c'est équivalent à zéro.



Nous avons donc remplacé les valeurs manquantes de ces variables par zéro.

- Variables discrètes: Pour ces jeu donner ce sont des variables binaire ou catégorielles qui ont des modalités de types numériques.

SELECTED COLUMNS

All Types ▾ search column

Converted  
Newsletter\_Subscription  
Account\_Creation  
SIC  
Source\_cpc  
Source\_direct  
Source\_email  
Source\_organic  
Source\_referral

Nous avons donc remplacer les valeurs manquantes de ces variables par leurs mode.

- Variables qualitatives de modalité de types Char (chaine de caractère) :

SELECTED COLUMNS

All Types ▾ search col

IndustrySector  
Employee\_Size  
Acquired\_Site\_Flag  
Sister\_Company\_Flag  
ContactDeptDesc  
Last\_Purchase\_Period  
Type\_visitor  
Top\_WebCat1  
Top\_WebCat2  
Top\_ProductName  
Top\_ProductFamily

Nous avons donc remplacer les valeurs manquantes de ces variables par NR (Non Renseigné)

### b. Remove Duplicate Rows

Ce composant quant à lui permet de supprimer les doublons potentiels d'un jeu de données. Pour identifier les doublons, on a sélectionner uniquement la colonne **Journey\_ID** comme colonne clé pour s'assurer que les identifiants soient uniques.

### c. Select Columns in Dataset

Composant de microsoft azure ML qui permet de choisir un sous-ensemble de colonnes à utiliser dans les opérations en aval. Ce composant ne supprime pas les colonnes du jeu de données source ; au lieu de cela, il crée un sous-ensemble de colonnes. Dans notre cas ça nous a permis de sélectionner les colonnes qu'on trouve pertinentes pour entraîner notre modèle. On a donc sélectionné toutes les colonnes sauf les colonnes [Journey\\_ID](#), [SIC](#) et [Nb\\_catalog\\_Requests](#). Ce sont ces trois dernières colonnes qui nous apparaissaient évident d'écarter. En effet on a décidé d'écarter la colonne [SIC](#) car elle apporte presque la même information que la colonne [IndustrySector](#), elles sont fortement corrélées, on a donc décidé de garder qu'une parmi les deux. Quant à la colonne [Journey\\_ID](#) on l'a pas choisi car n'apporte pas d'information sur notre variable cible donc ne l'explique pas. La variable [Nb\\_catalog\\_Requests](#) n'a pas été sélectionnée car cette colonne a beaucoup de valeurs manquantes, elle est presque vides.

### d. Clip Values

On utilise le composant Clip Values pour identifier et éventuellement remplacer les valeurs de données qui sont supérieures ou inférieures à un seuil spécifié par une moyenne, une constante ou une autre valeur de substitution. On l'applique uniquement sur nos variables quantitatives. On a choisi l'option « ClipPeaksAndSubpeaks » pour pouvoir spécifier à la fois les limites supérieures et inférieures.

### e. Edit Metadata

Le composant [Edit Metadata](#) est utilisé pour modifier les métadonnées associées aux colonnes d'un jeu de données. La valeur et le type de données du jeu de données changeront après l'utilisation du composant [Edit Metadata](#). Dans notre cas on l'a utilisé pour convertir les variables de modalité de type Char ( chaîne de caractère) en données catégorielles.

### f. Filter Based Feature Selection

Ce composant nous a aidé à identifier les colonnes de notre jeu de données d'entrée qui ont le plus grand pouvoir prédictif. On a donc choisi les 15 variables qui expliquent mieux la variable cible, qui ont un pouvoir prédictif significatif. Pour se faire on s'est servi des corrélations entre les variables explicatives et la variable cible, variable à expliquer (Converted). Les autres variables étant faiblement corrélées avec la variables cibles sont

écarter par le composant « **Filter Based Feature Selection** ». Avec les 15 variables nous avons la même précision que quand on avait utilisé toutes les variables.

#### g. Split Data

On a utilisé le composant « Split Data » pour diviser notre jeu de données en deux ensembles distincts. On a utilisé l'option **Split Rows** pour diviser les données en deux parties : ensemble d'entraînement et de test . J'ai pris 50% des données pour l'ensemble de test et 50% autre pour l'ensemble d'entraînement.

## B. Entraînement du modèle

Compte tenu des différentes variables online et offline on doit déterminer si la Customer Journey se terminera par une conversion ou non. Notre variable cible (Converted) est donc binaire , nous sommes donc dans une situation de classification. On a donc décidé de travailler avec deux algorithmes de classification Two-Class Logistic Regression (régression logistique à deux classes) et Two-Class Boosted Decision Tree (l'arbre de décision boosté à deux classes)

#### a. Two-Class Logistic Regression

La régression logistique est une technique qui est utilisée pour modéliser de nombreux types de problèmes. Cet algorithme est une méthode *d'apprentissage supervisé*. On a donc utilisé ce premier algorithme d'apprentissage supervisé pour apprendre notre modèle.

#### b. Two-Class Boosted Decision Tree

Ce composant est utilisé pour créer un modèle d'apprentissage automatique basé sur l'algorithme des arbres de décision boostés. On a donc utilisé ce second *algorithme d'apprentissage pour apprendre notre modèle*.

#### c. Train Model

Cette composante est utilisée pour former un modèle de classification ou de régression. Comme dans notre cas nous sommes dans une situation de classification , nous l'avons utilisé pour entraîner chacun de nos deux modèles de classification.

#### d. Score Model

On a utilisé ce composant pour générer des prédictions à l'aide de nos deux modèles de classification entraînés.










**e. Evaluate Model**

Ce composant permet de mesurer la précision d'un modèle entraîné. On s'est donc servi de ce composant pour trouver la précision (accuracy) de chacun de nos deux modèles. Cela permet donc de déterminer si notre modèle est bon ou non.

**f. Execute R Script**

Le composant « Execute R » Script permet d'exécuter le code R dans le concepteur Azure Machine Learning. On s'est donc servi de ce composant pour sélectionner les valeurs de score de précision de nos modèles créés à partir des sorties de « Evaluate Model ». On a donc pu comparer les deux modèles facilement.

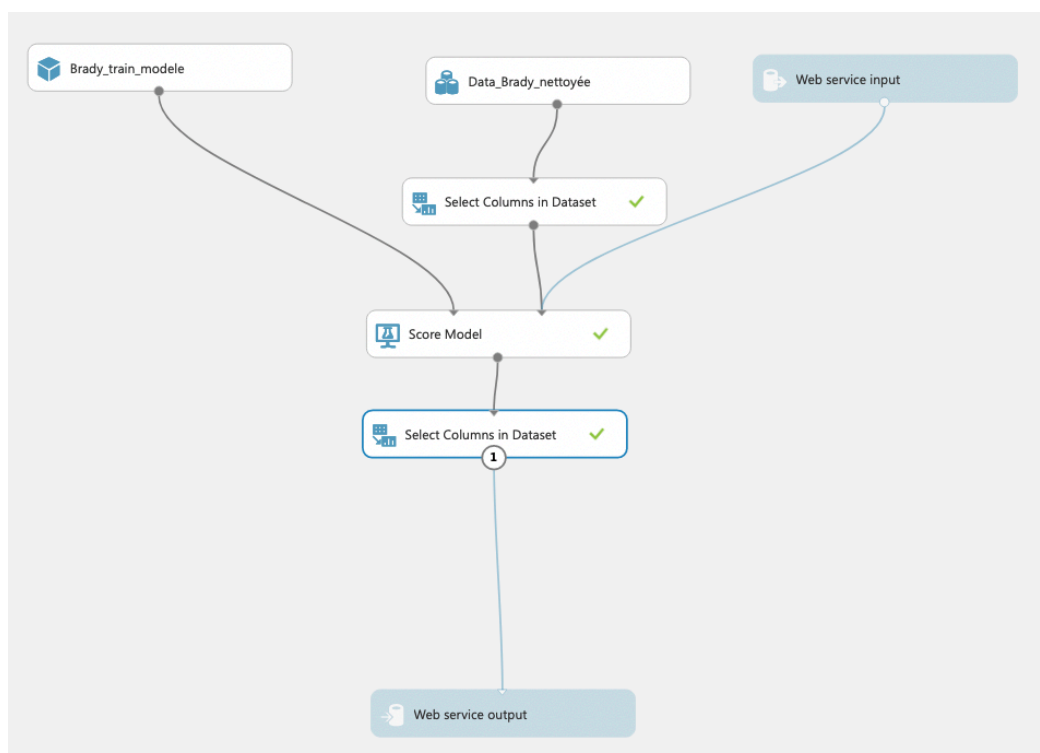
**g. Comparaison des deux modèles entraînés**

	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss	Converted
view as 								
	0.956694	0.94657	0.953272	0.949909	0.987702	0.145873	78.658777	Regression logistic
	0.974121	0.963861	0.976535	0.970157	0.995978	0.078807	88.470474	Arbre de decision

L'arbre de de décision a la plus forte précision (97,4%), c'est donc le meilleurs modèle pour ce jeu de données. C'est donc avec ce modèle que nous allons utiliser pour faire le déploiement.

**2. Créer une 2<sup>ème</sup> expérience qui servira au déploiement du meilleur modèle choisi en tant que web service afin de prédire la conversion d'une customer journey en fonction de nouvelles valeurs des variables choisies.**

Dans cette partie nous avons créer une nouvelle expérience pour le déploiement de notre modèle que nous avons nommé « [Projet Déploiement du modèle Brady](#) ». Nous avons ensuite enregistré notre base de donnée nettoyée nommé [Data\\_Brady\\_nettoyée](#) et notre meilleur modèle entraîné nommé [Brady\\_train\\_modele \(Two-Class Boosted Decision Tree\)](#) obtenus à partir de l'expérience précédente puis nous les avons chargés dans cette nouvelle expérience. Nous avons ensuite utilisé le composant « Select Columns in Dataset » pour sélectionner que les variables explicatives puis nous avons ensuite utilisé le composant Score pour générer des prédictions. Nous avons par la suite utilisés une seconde fois le composant « Select Columns in Dataset » pour n'afficher que la colonne « [Scored Labels](#) » prédit par le modèle. Nous avons en fin déployer le modèle en tant que web service afin de prédire la conversion d'une customer journey en fonction de nouvelles valeurs des variables choisies.



### 3. Publier le modèle en tant que web service et l'utiliser avec RStudio.

Vous trouver le fichier du déploiement rstudio dans les fichier joints

## **II - définir des groupes de customer journeys ayant les mêmes caractéristiques et affinités de comportement**

Nous avons décidé de faire cette partie directement dans la première expérience intitulée « [Projet Brady](#) ». Pour classer les customer journeys en différents clusters ayant les mêmes caractéristiques on a utilisé le modèle d'apprentissage non supervisé K-Mean.

**4. Créer une nouvelle expérience afin de classer les customer journeys en différents clusters ayant les mêmes caractéristiques. Pour cela, vous pourrez utiliser le modèle d'apprentissage non supervisé K-Means.**

### **a. Normalize Data**

Comme nos variables n'ont pas la même échelle, elle n'ont pas la même unité de mesure, on a donc décidé de les normaliser avec le composant « Normalize Data ». L'objectif est de modifier les valeurs des colonnes numériques dans l'ensemble de données pour utiliser une échelle commune, sans déformer les différences dans les plages de valeurs ou perdre des informations.

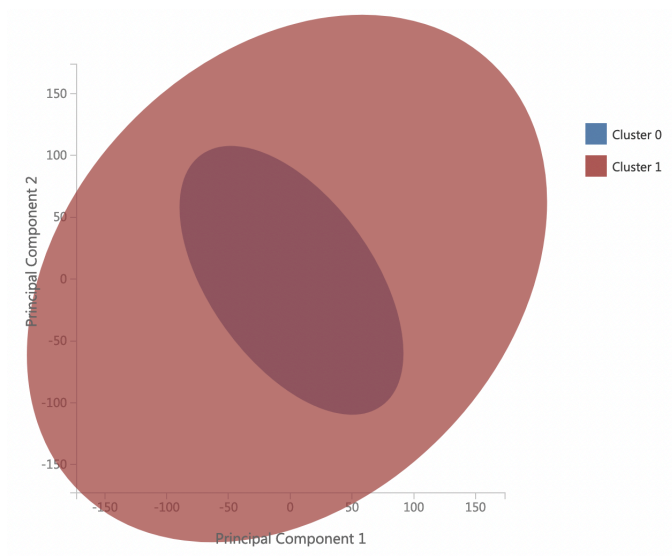
### **b. Train Clustering Model**

On a utilisé ce composant pour former un modèle de regroupement.



### c. Assign Data to Clusters

Le composant génère des prédictions à l'aide d'un modèle de clustering qui a été entraîné avec l'algorithme de *clustering K-means*.



Le modèle a donc déterminé deux classes (groupe) de même caractéristiques pour ce jeux de données.

### d. Convert to Dataset

Permet de convertir la sortie du composant « Assign Data to Clusters » en dataset.

## La structure finale de la partie II

