

# Projet n°5 : “Segmentation Client”

Soutenance de Jury - Projet n°5 - Octobre 2021  
Ali Naama

# Sommaire



**I. Description de la problématique**

**II. Analyse exploratoire du jeu de données**

**III. Analyse détaillée et observations**

**IV. Résultats**

**V. Conclusions et recommandations**

# I. Description de la problématique et exploration du jeu de données

# Description de la problématique



## Description :

Un client d'une grande entreprise française souhaiterait cibler ses clients à partir des données de ventes annuelles.

Il dispose pour cela des données de ventes de l'exercice 2020 issue d'une application CRM Salesforce Et du module Sales Cloud.

Les données de ventes qui seront à notre disposition sont :

N° de Contrat, N° de Client, Produit, Quantité, Date de Vente, Prix unitaire

La segmentation sera construite à partir des nouvelles variables suivantes :

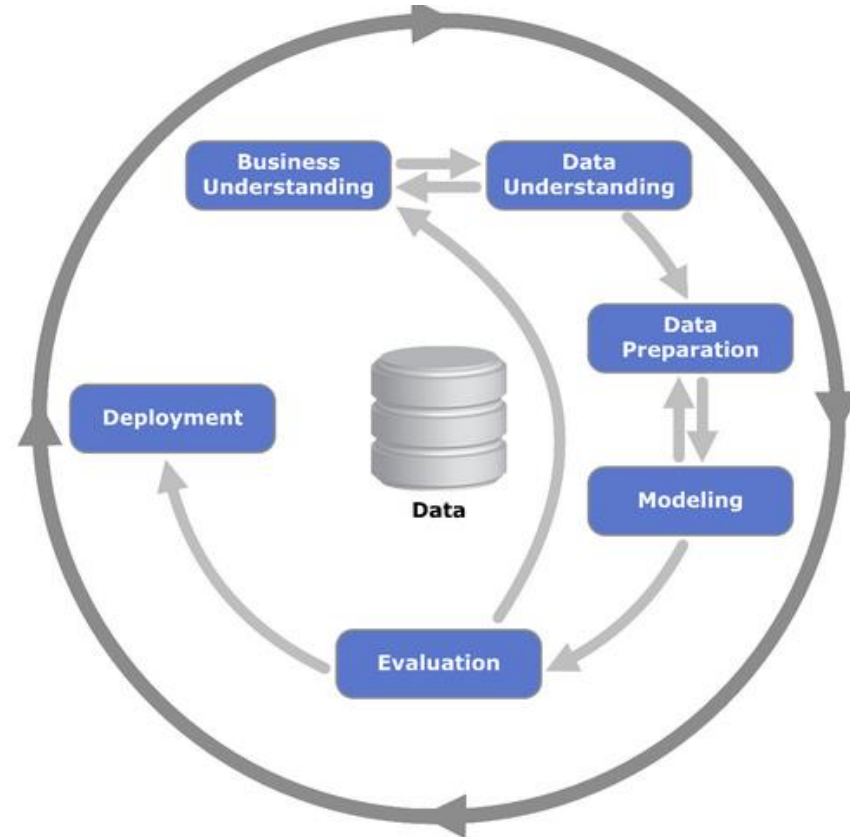
**Total des ventes, Nombre de Commandes, Moyenne d'achat des Commandes**

## Objectifs :

A partir des données de l'application CRM construire une segmentation client à l'aide des algorithmes « K-means » et mettez en œuvre la reprise des données clients afin que les utilisateurs de l'application bénéficient des données pour mieux répondre aux clients.

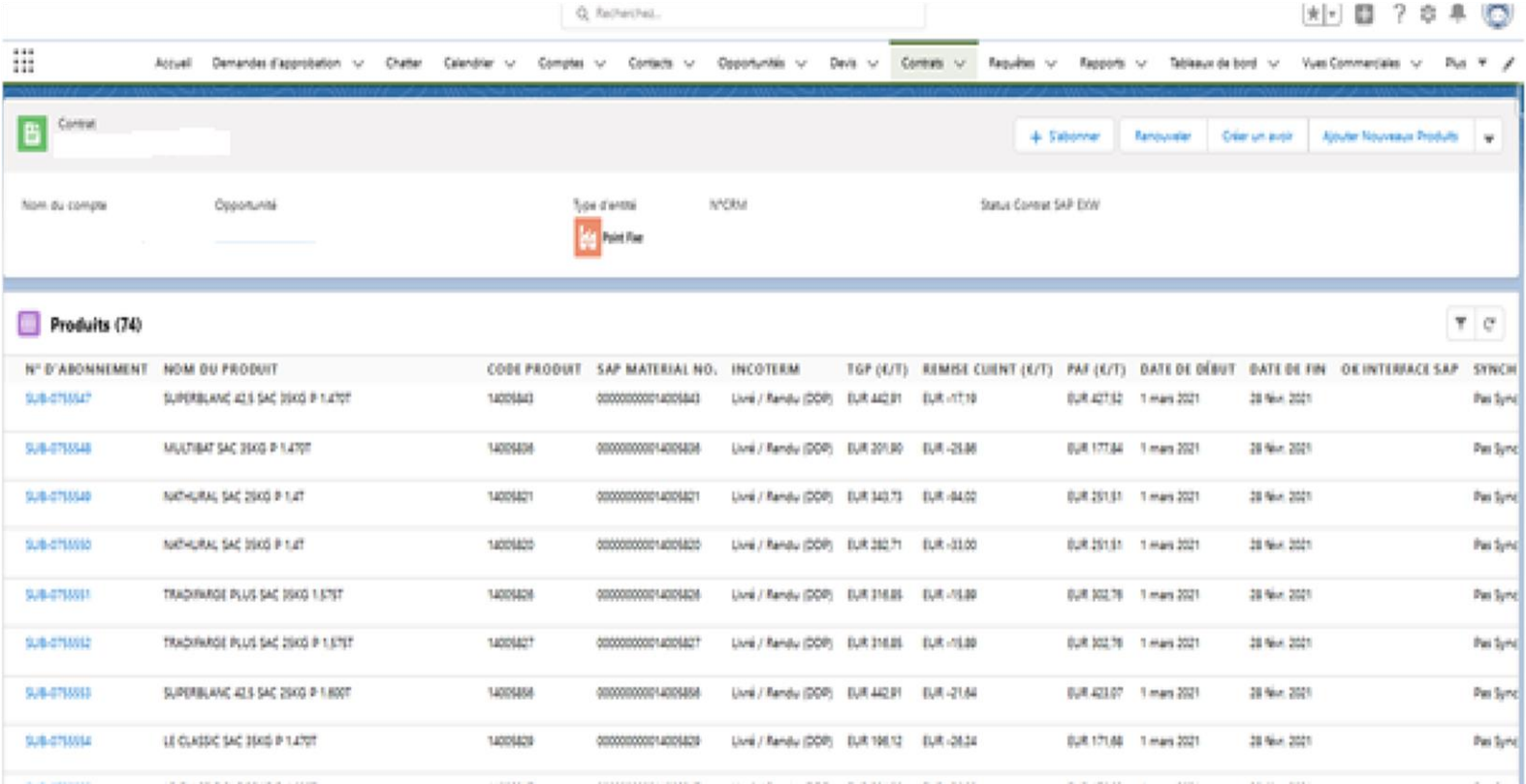
Pour rappel « **K-means** » est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents.

# Méthodologie de Data Science « CRISP DM »



# Exemple de Produits vendus – Salesforce CRM

Contrats



The image shows a screenshot of the Salesforce CRM interface. At the top, there is a navigation bar with various tabs: Accueil, Demandes d'approbation, Chatter, Calendrier, Comptes, Contacts, Opportunités, Devis, Contrats (selected), Requêtes, Rapports, Tableaux de bord, Vues Commerciales, and Plus. Below the navigation bar, there is a section for 'Contrat' with a search bar and buttons for '+ S'abonner', 'Renouveler', 'Créer un autre', and 'Ajouter Nouveaux Produits'. The main content area displays a table of products under the contract. The table has columns for N° d'abonnement, Nom du produit, Code produit, SAP Material No., Incoterm, TGP (€/T), Remise client (€/T), PAF (€/T), Date de début, Date de fin, OK Interface SAP, and Sync. The table lists 74 products, with the first 8 visible in the screenshot.

N° d'abonnement	Nom du produit	Code produit	SAP Material No.	Incoterm	TGP (€/T)	Remise client (€/T)	PAF (€/T)	Date de début	Date de fin	OK Interface SAP	Sync
SUB-0715547	SUPERBLANC 42,5 SAC 25KG P 1,470T	1402543	0000000001402543	Livré / Rendu (DOP)	EUR 442,81	EUR -17,19	EUR 425,62	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715548	MULTIBAT SAC 25KG P 1,470T	1402546	0000000001402546	Livré / Rendu (DOP)	EUR 201,80	EUR -23,80	EUR 177,99	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715549	NATHURAL SAC 25KG P 1,47	1402547	0000000001402547	Livré / Rendu (DOP)	EUR 143,73	EUR -84,00	EUR 59,73	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715550	NATHURAL SAC 25KG P 1,47	1402548	0000000001402548	Livré / Rendu (DOP)	EUR 282,71	EUR -31,00	EUR 251,71	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715551	TRACHIRAOE PLUS SAC 25KG P 1,575T	1402549	0000000001402549	Livré / Rendu (DOP)	EUR 218,85	EUR -15,89	EUR 202,96	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715552	TRACHIRAOE PLUS SAC 25KG P 1,575T	1402550	0000000001402550	Livré / Rendu (DOP)	EUR 218,85	EUR -15,89	EUR 202,96	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715553	SUPERBLANC 42,5 SAC 25KG P 1,800T	1402551	0000000001402551	Livré / Rendu (DOP)	EUR 442,81	EUR -21,64	EUR 421,17	1 mars 2021	28 Nov. 2021		Pas Sync
SUB-0715554	LE CLASSIC SAC 25KG P 1,470T	1402552	0000000001402552	Livré / Rendu (DOP)	EUR 198,12	EUR -28,14	EUR 170,00	1 mars 2021	28 Nov. 2021		Pas Sync

Souscriptions  
annuelles liées

# Présentation du jeu de données



## Informations sur les variables :

- ☐ Les données sont extraites de l'application CRM via un pipeline Python automatisé qui permet la restitution sous format Excel
- ☐ Les colonnes ont été choisies directement à partir des données de ventes nommées : Contrat / Ligne de Souscription annuelle

Nombre de colonnes du fichier : 12

Nombre de lignes du fichier : 287879

# Variables du jeu de données



Champ	Valeur d'exemple	Type	Description
Name	SUB-0542053	Texte	Identifiant unique de la souscription liée au Contrat
SBQQ__ProductName__c	YYYPRD45454545	Texte	Nom du Produit souscrit
SBQQ__Quantity__c	1	Flottant	Quantité du Produit souscrit
Description__c	YUTREGENA3523	Texte	Description
Code_Produit__c	14005858LKI	Texte	Code Produit
SBQQ__Account__c	21221002M4luYAAR	Texte	Identifiant du Client
SBQQ__Contract__c	8006900000AnAntAAN	Texte	Identifiant du Contrat
SBQQ__ContractNumber__c	CON-000171454	Texte	numéro de Contrat
SBQQ__StartDate__c	01/01/2020	Date	Date de Début de Contrat
SBQQ__EndDate__c	30/06/2020	Date	Date de Fin de Contrat
SBQQ__NetPrice__c	223.6	Flottant	Prix Net
SBQQ__Contract__r.BillingCountry	France	Texte	Pays de Facturation du Contrat

Nombre de colonnes du fichier : 12

Nombre de lignes du fichier : 287879



# Pipeline d'extraction – API Salesforce Bulk



## Extraction via l'API Bulk Salesforce en Python

### Étapes :

- ❑ Requêtes : SOQL Salesforce via API Bulk
- ❑ Extraction, Transformation : Format Json à Dataframe panda
- ❑ Sauvegarde au format Excel via panda

```
from simple_salesforce import Salesforce, SalesforceLogin, SFType
from salesforce_bulk import SalesforceBulk
from salesforce_bulk.util import IteratorBytesIO
from urllib.parse import urlparse
from salesforce_bulk import CsvDictsAdapter
import numpy as np

# Log into sf / Connexion to SF
login_results = client.login()
print(login_results)

# 0.1 Log mgmt
#
now = datetime.datetime.now()
logging.basicConfig(filename='D:\Projet Perso\All\Data Scientist\Projet 5\log\ExtractSF.log', level=logging.INFO)
logging.info('Started : ' + now.strftime('%Y-%m-%d %H:%M:%S'))

query = "select Name, SBQQ__ProductName__c, SBQQ__Quantity__c, Description__c, Code-Produit__c, SBQQ__Account__c, SBQQ__Contract__c, SBQQ__ContractNumber__c, \
        SBQQ__StartDate__c, SBQQ__EndDate__c, SBQQ__NetPrice__c, SBQQ__Contract__c.BillingCountry from SBQQ__Subscription__c where ' \
        CreatedDate >= 2020-01-01T00:00:00Z and CreatedDate < 2021-01-01T00:00:00Z order by SBQQ__Contract__c, CreatedDate desc"

# Data Migration - User - Quality
sf = Salesforce(organizationId='[redacted]', username='[redacted]', password='[redacted]', proxy='[redacted]')
data = sf.bulk.SBQQ__Subscription__c.query_all(query)
#dataframe = pd.DataFrame(data['records'])

df = pd.DataFrame.from_dict(data, orient='columns').drop('attributes', axis=1)
writer = pd.ExcelWriter('D:\Projet Perso\All\Data Scientist\Projet 5\data\ExtractSales2020.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sales', index=False)
# Close the Pandas Excel writer and output the Excel file.
writer.save()
```

# Présentation du jeu de données



Fichier	Nb lignes	Nb colonnes	Taux remplissage moyen	Doublons	Description
2020AnnualSales.xlsx	287879	12	100.0%	0	Fichier des ventes 2020

- ❑ On notera que l'on dispose d'un excellent taux de remplissage des données
- ❑ On notera que la colonne Quantité dispose d'une valeur négative : [ 1 , -1, 190]

```
# Analyse unic values of Data Frame :  
  
# Get unique values in column 'SBQQ__Quantity__c' of the dataframe  
uniqueValues = df['SBQQ__Quantity__c'].unique()  
print('Unique elements in column "SBQQ__Quantity__c" ' )  
print(uniqueValues)
```

# Outils utilisés pour l'analyse



Nom	Utilisation	Fonctions spécifiques
<b>PyCharm 2021.1</b>	Test et développement	IDE Community Edition, Debug, Synchro Git
<b>Python 3.9.5</b>	Moteur Python Gestionnaire de librairies	Moteur d'exécution
<b>Pandas 1.4.0</b>	Librairie de manipulation de données Représentation des données	Manipulation de Dataframe : création, copie, filtres, tris, description, concaténation, pivotage, autre
<b>Matplotlib 3.5.1</b> <b>Seaborn 0.11.2</b> <b>Numpy 1.22.0</b>	Génération de graphiques Gestion des densités de probabilité	Barplot, Scatterplot, lineplot, distplot, heatmap Calcul statistique



## II. Analyse Exploratoire

# Stratégie d'analyse



1

**Découvrir les  
Données**

Analyse des données des 2  
fichiers

Données  
manquantes ?



2

**Analyse  
Exploratoire /  
Feature  
Engineering**

Travaux de Nettoyage et de  
feature engineering  
K-Means  
Clustering



3

**Comparer / Tester**

Sélection du bon  
nombre de  
Clusters



4

**Interprétation /  
Validation**

Interprétation des  
Résultats

Validation Métier

Reprise de Données  
CRM Salesforce

# III. Analyse

# Feature Engineering

- ❑ Nous commençons par exclure les quantités négatives ne correspondant pas aux ventes annuelles
- ❑ Nous créons les nouvelles variables nécessaires à l'analyse à savoir :
- ❑ Total des Ventes: **TotalSales** qui vaut **la Quantité \* Prix**
- ❑ Nombre de Ventes : Pour chaque enregistrement qui représente l'historique des ventes nous allons agréger les données par client (SBQQ\_\_Account\_\_c) et par lignes de souscriptions (champ Name qui est unique)
- ❑ Nous créons 3 variables ensuite via l'agrégation par client et par souscriptions ce qui nous donne la Moyenne des ventes, le Total des Ventes et le nombre des ventes

```
# 2. Data Clean-Up
df.loc[df['SBQQ__Quantity__c'] <= 0].shape
print(df.shape)

#### - Total Sales
df['Sales'] = df['SBQQ__Quantity__c'] * df['SBQQ__NetPrice__c']

print(df.head())

#### - Per Customer Data
customer_df = df.groupby('SBQQ__Account__c').agg({
    'Sales': sum,
    'Name': lambda x: x.nunique()
})

#### - Total Sales
customer_df.columns = ['TotalSales', 'OrderCount']
customer_df['AvgOrderValue'] = customer_df['TotalSales']/customer_df['OrderCount']
```

# Feature Engineering

- ❑ L'agrégation des ventes par identifiant Client et Nom de Souscription nous donne le tableau suivant avec les variables à analyser

SBQQ__Account__c	TotalSales	OrderCount	AvgOrderValue
12122M4hxDAAR	5534.39	58	95.420517
333330002M4hxEAAR	5791.43	62	93.410161
12120002M4hxFAAR	1546.22	15	103.081333
1548702M4hxGAAR	4589.00	52	88.250000
487002M4hxLAAR	1737.70	14	124.121429

- ❑ Dans la mesure où les algorithmes de clustering sont très sensibles à l'échelle des données, nous allons normaliser nos données via une méthode de ranking :

```
#### - Ranking
rank_df = customer_df.rank(method='first')
print(rank_df.head(15))

#### - Normalize
normalized_df = (rank_df - rank_df.mean()) / rank_df.std()
```

SBQQ__Account__c	TotalSales	OrderCount	AvgOrderValue
0011r00002M4hxDAAR	5085.0	5785.0	1987.0
0011r00002M4hxEAAR	5188.0	5907.0	1718.0
0011r00002M4hxFAAR	1678.0	2239.0	3010.0
0011r00002M4hxGAAR	4859.0	5581.0	1258.0
0011r00002M4hxLAAR	3019.0	1826.0	5169.0
0011r00002M4hxMAAR	1239.0	1100.0	4256.0
0011r00002M4hxOAAR	4452.0	4291.0	3631.0

SBQQ__Account__c	TotalSales	OrderCount	AvgOrderValue
0011r00002M4hxDAAR	0.541468	0.854456	-0.843725
0011r00002M4hxEAAR	0.587522	0.909005	-0.964002
0011r00002M4hxFAAR	-0.981887	-0.731050	-0.386310
0011r00002M4hxGAAR	0.440418	0.763243	-1.169679



# Statistiques

❑ Statistique de notre échantillon normalisé :

```
#### - Normalize
normalized_df = (rank_df - rank_df.mean()) / rank_df.std()

print(normalized_df.head(15))
print(normalized_df.describe())
```

	TotalSales	OrderCount	AvgOrderValue
count	7747.000000	7.747000e+03	7.747000e+03
mean	0.000000	-5.869980e-17	-7.337475e-18
std	1.000000	1.000000e+00	1.000000e+00
min	-1.731715	-1.731715e+00	-1.731715e+00
25%	-0.865858	-8.658577e-01	-8.658577e-01
50%	0.000000	0.000000e+00	0.000000e+00
75%	0.865858	8.658577e-01	8.658577e-01
max	1.731715	1.731715e+00	1.731715e+00

# K-means clustering

- ❑ L'algorithme de k-means clustering est utilisé afin de représenter les groupes de données similaires.
- ❑ Pour cela nous allons utiliser la librairie : scikit-learn en lui indiquant que nous voulons construire nos segments sur les 3 variables : 'TotalSales', 'OrderCount', 'AvgOrderValue'
- ❑ Le principe est que l'on présuppose que l'on a par exemple 4 clusters (segments) et voir si c'est la meilleure segmentation de nos données. Le modèle est dit entraîné avec ces premiers paramètres (fonction « fit »).
- ❑ L'objet kmeans stockera les libellées et centres de nos segments :

```
from sklearn.cluster import KMeans

#### - K-Means Clustering
kmeans = KMeans(n_clusters=4).fit(normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']])
print(kmeans)
kmeans.labels_
kmeans.cluster_centers_

four_cluster_df = normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']].copy(deep=True)
four_cluster_df['Cluster'] = kmeans.labels_
four_cluster_df.head()

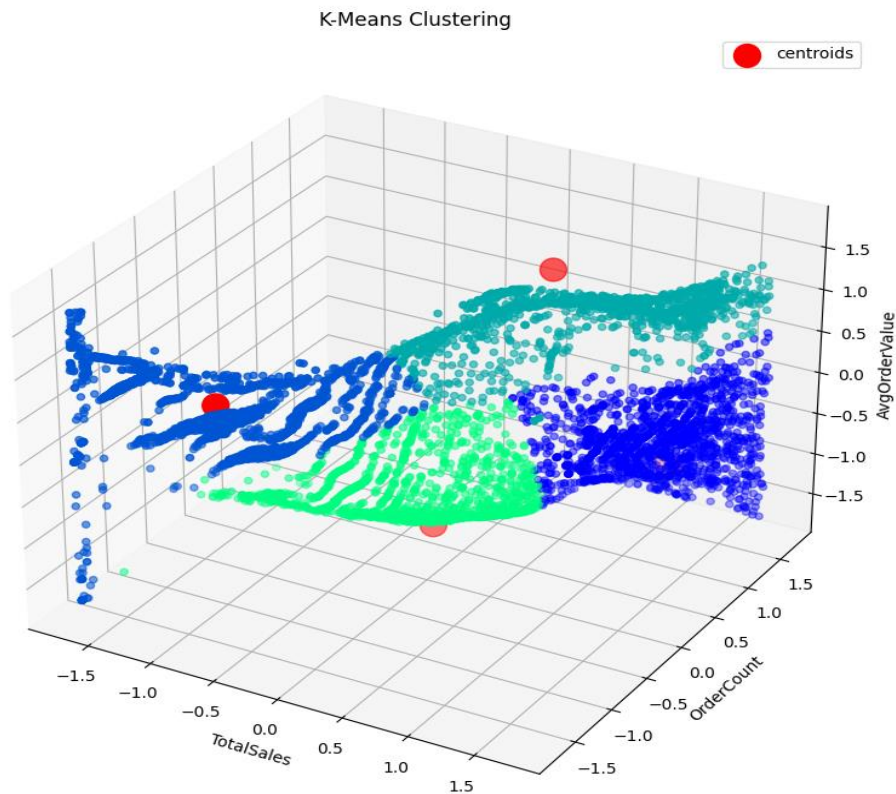
four_cluster_df.groupby('Cluster').count()['TotalSales']
```

# K-means clustering - 3D

```
## 3 D Printing
```

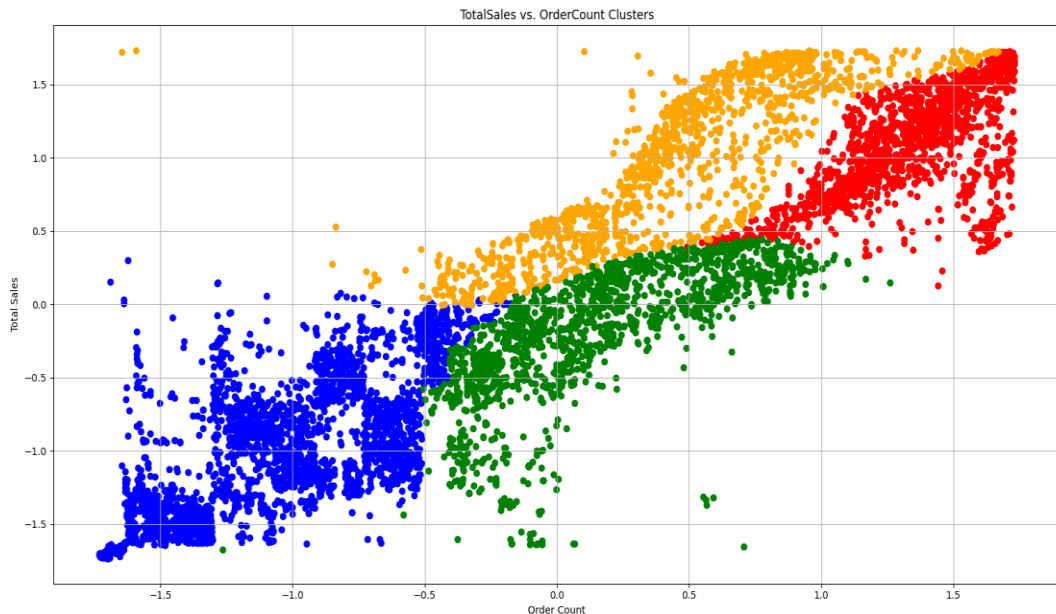
```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
clusts = KMeans(n_clusters=4).fit_predict(normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']])
#Plot the clusters obtained using k means
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(
    kmeans.cluster_centers[:, 1],
    kmeans.cluster_centers[:, 0],
    kmeans.cluster_centers[:, 2],
    s = 250,
    marker='o',
    c='red',
    label='centroids')
scatter = ax.scatter(four_cluster_df['TotalSales'], four_cluster_df['OrderCount'], four_cluster_df['AvgOrderValue'],
                    c=clusts, s=20, cmap='winter')

ax.set_title('K-Means Clustering')
ax.set_xlabel('TotalSales')
ax.set_ylabel('OrderCount')
ax.set_zlabel('AvgOrderValue')
ax.legend()
plt.show()
```



# K-means clustering

- ❑ Nous pouvons visualiser grâce à ce graphique que le cluster en bleu représente les clients ayant le total des achats et le nombre de souscriptions le moins important.
- ❑ Le cluster en rouge est au contraire le plus performant en nombre de souscriptions réalisés et total d'achat.



```
plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['TotalSales'],
    c='blue'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['TotalSales'],
    c='red'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['TotalSales'],
    c='orange'
)

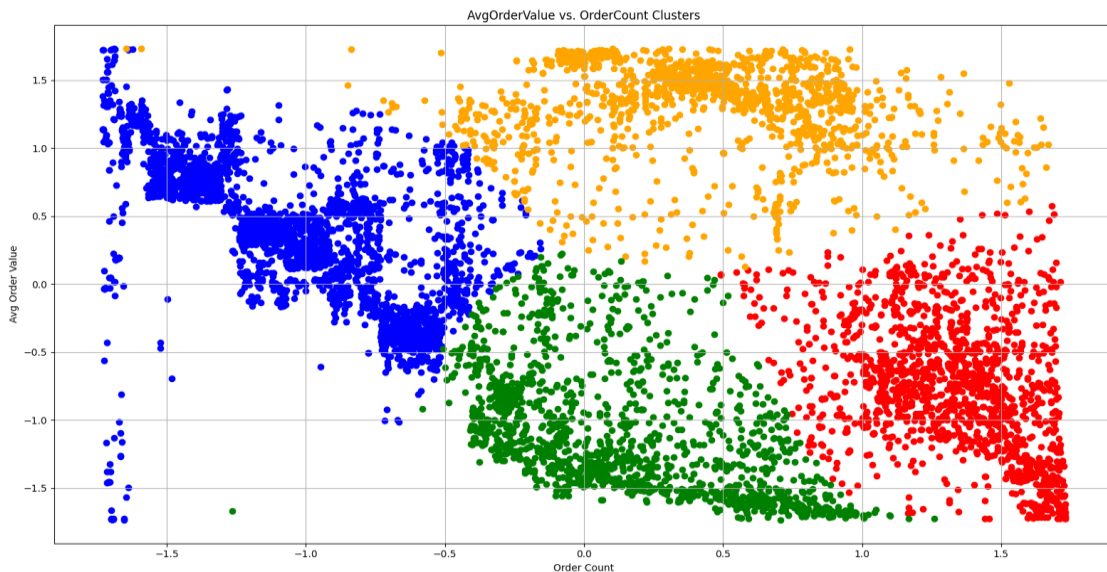
plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['TotalSales'],
    c='green'
)

plt.title('TotalSales vs. OrderCount Clusters')
plt.xlabel('Order Count')
plt.ylabel('Total Sales')

plt.grid()
plt.show()
```

# K-means clustering

- ❑ On visualise ici une segmentation pour les 2 variables Moyenne des souscriptions réalisées avec la variable Nombre de souscriptions réalisés.
- ❑ Le cluster en Bleu possède la moyenne d'achat la plus haute avec le nombre de souscription le plus bas.
- ❑ Au contraire le cluster en rouge possède le nombre de souscription le plus élevé et la moyenne d'achat la plus basse



```
plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['AvgOrderValue'],
    c='blue'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['AvgOrderValue'],
    c='red'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['AvgOrderValue'],
    c='orange'
)

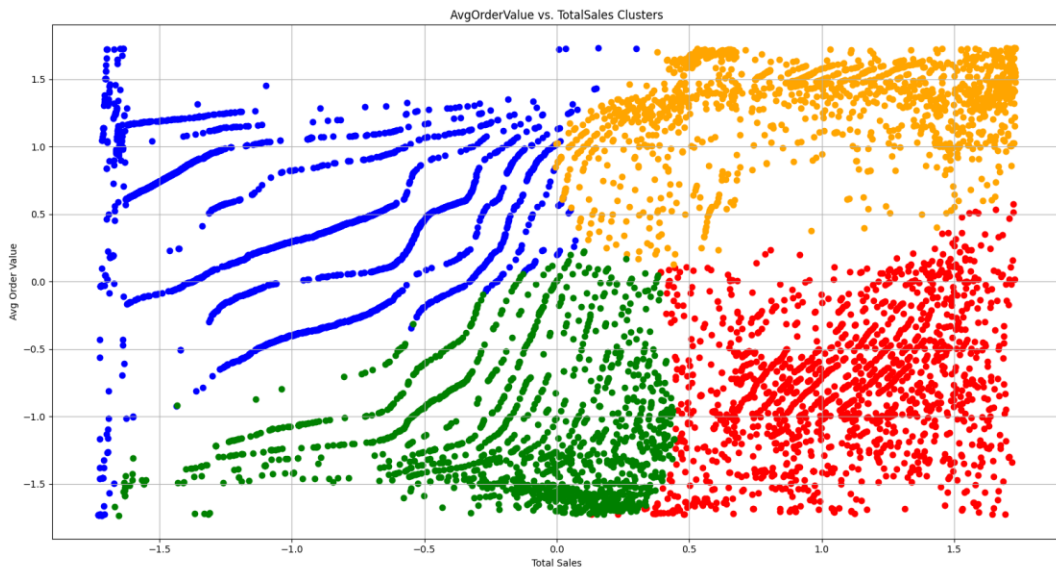
plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['OrderCount'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['AvgOrderValue'],
    c='green'
)

plt.title('AvgOrderValue vs. OrderCount Clusters')
plt.xlabel('Order Count')
plt.ylabel('Avg Order Value')

plt.grid()
plt.show()
```

# K-means clustering

- ❑ On visualise ici une segmentation pour les 2 variables Moyenne des montants de souscriptions et Total des achats réalisés.
- ❑ En bleu le cluster avec le Total des souscriptions le plus bas Et la moyenne des achats la plus haute.
- ❑ Le cluster en rouge avec le cluster opposé : Total Sales le plus haut et Moyenne des achats la plus basse.



```
plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['TotalSales'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 0]['AvgOrderValue'],
    c='blue'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['TotalSales'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 1]['AvgOrderValue'],
    c='red'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['TotalSales'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 2]['AvgOrderValue'],
    c='orange'
)

plt.scatter(
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['TotalSales'],
    four_cluster_df.loc[four_cluster_df['Cluster'] == 3]['AvgOrderValue'],
    c='green'
)

plt.title('AvgOrderValue vs. TotalSales Clusters')
plt.xlabel('Total Sales')
plt.ylabel('Avg Order Value')

plt.grid()
plt.show()
```

# Sélection du Nombre de « Clusters »

- ❑ Au départ nous ne connaissons pas le nombre optimal de cluster. En revanche nous pouvons utiliser le score « silhouette »

```
Silhouette Score for 4 Clusters: 0.4974
Silhouette Score for 5 Clusters: 0.4551
Silhouette Score for 6 Clusters: 0.4428
Silhouette Score for 7 Clusters: 0.4179
Silhouette Score for 8 Clusters: 0.4028
```

- ❑ On mesure pour chaque valeur d'hypothèse de 4 à 8, la valeur du score silhouette afin de déterminer le score le plus bas.
- ❑ Dans notre cas, on confirme que le score le plus bas est le cluster 4, ce qui nous permettra maintenant d'expliquer nos résultats.

```
from sklearn.metrics import silhouette_score

for n_cluster in [4, 5, 6, 7, 8]:
    kmeans = KMeans(n_clusters=n_cluster).fit(
        normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']]
    )
    silhouette_avg = silhouette_score(
        normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']],
        kmeans.labels_
    )

    print('Silhouette Score for %i Clusters: %0.4f' % (n_cluster, silhouette_avg))
```

# Sélection du Nombre de « Clusters »

- ❑ Au départ nous ne connaissons pas le nombre optimal de cluster. En revanche nous pouvons utiliser le score « silhouette » via la librairie sklearn :

```
Silhouette Score for 4 Clusters: 0.4974
Silhouette Score for 5 Clusters: 0.4551
Silhouette Score for 6 Clusters: 0.4428
Silhouette Score for 7 Clusters: 0.4179
Silhouette Score for 8 Clusters: 0.4028
```

- ❑ On mesure pour chaque valeur d'hypothèse de 4 à 8, la valeur du score silhouette afin de déterminer le score le plus bas.
- ❑ Dans notre cas, on confirme que le score le plus bas est le cluster **4**, **ce qui nous permettra maintenant d'interpréter nos résultats.**

```
from sklearn.metrics import silhouette_score

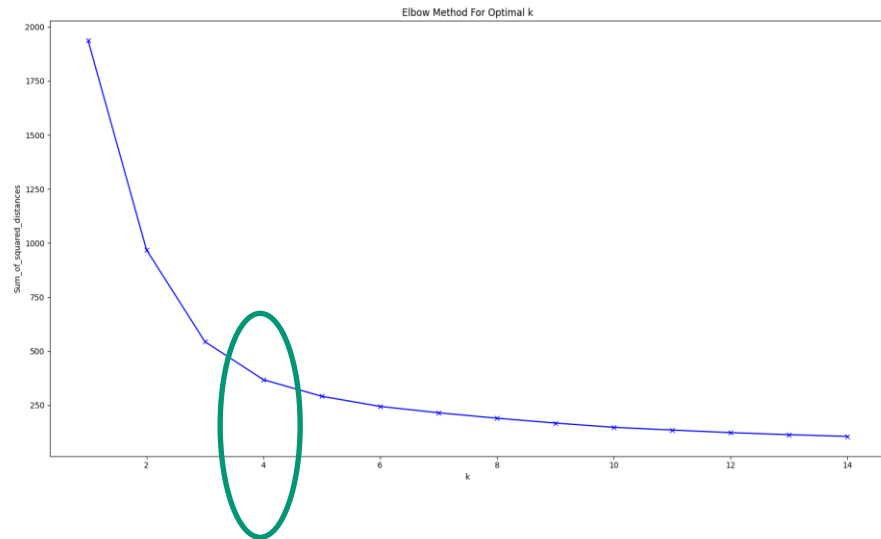
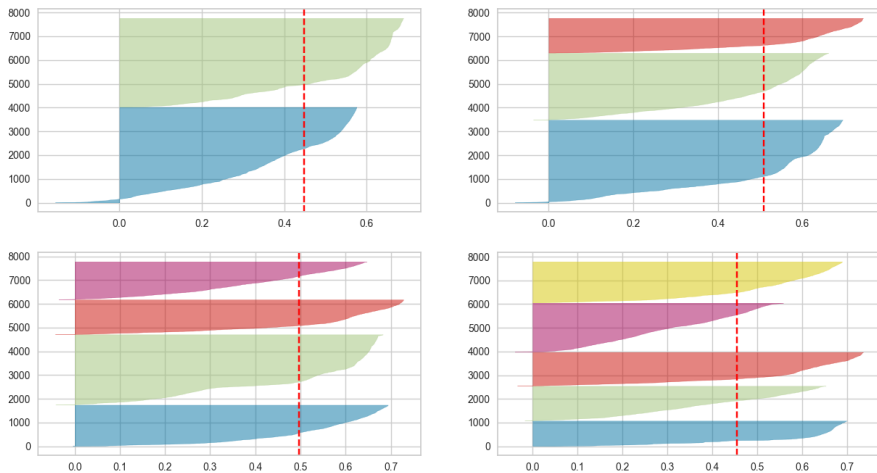
for n_cluster in [4, 5, 6, 7, 8]:
    kmeans = KMeans(n_clusters=n_cluster).fit(
        normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']]
    )
    silhouette_avg = silhouette_score(
        normalized_df[['TotalSales', 'OrderCount', 'AvgOrderValue']],
        kmeans.labels_
    )

    print('Silhouette Score for %i Clusters: %0.4f' % (n_cluster, silhouette_avg))
```



# Sélection du Nombre de « Clusters » optimale

☐ Méthode Silhouette et Elbow



# V. Résultats et recommandations

# Interprétation de notre « Segmentation Client »

- ❑ Il est intéressant de visualiser le centre de nos clusters :

```
[[ 0.95702415  0.43022394  1.31062639]
 [-0.98983024 -1.06677434  0.40101598]
 [ 1.04755822  1.31045336 -0.76958079]
 [-0.18445047  0.15509936 -1.10758077]]
```

SBQQ__Account__c	TotalSales	OrderCount	AvgOrderValue	Cluster
0011r00002M4hxDAAR	0.541468	0.854456	-0.843725	2
0011r00002M4hxEAAR	0.587522	0.909005	-0.964002	2
0011r00002M4hxFAAR	-0.981887	-0.731050	-0.386316	1
0011r00002M4hxGAAR	0.440418	0.763243	-1.169679	3
0011r00002M4hxLAAR	-0.382292	-0.915712	0.579027	1
0011r00002M4hxNAAR	-1.178175	-1.240325	0.170802	1
0011r00002M4hxOAAR	0.258438	0.186451	-0.108651	3
0011r00002M4hxPAAR	0.260227	0.134138	0.168119	3
0011r00002M4hxSAAR	-0.482001	-0.322824	-0.894697	3

- ❑ Les clients appartenant au cluster n°2 sont des clients faisant des achats fréquemment. Il pourra donc être pertinent de cibler cette population avec des produits à prix bas.
- ❑ Les clients appartenant au cluster n°1 sont des clients ayant une contribution au revenu et au nombre de souscription qui est moyenne. Par contre la moyenne d'achat par souscription est élevée. Ils achètent des produits chers fréquemment. Il conviendra donc, de continuer à les cibler avec des produits à prix élevé.

# Interprétation de notre « Segmentation Client »

Cluster (Ligne de la matrice)	Description	Ciblage et actions Métier à prévoir
0	'TotalSales' et 'OrderCount' élevé mais 'AvgOrderValue' très bas	Amener ces clients à augmenter le volume d'achat en les sollicitant plus (mailing, etc)
1	'TotalSales' et 'OrderCount' très bas mais 'AvgOrderValue' médian	Amener ces clients à augmenter la fréquence d'achat en les sollicitant plus (mailing, etc)
2	'TotalSales' élevé , 'OrderCount' médian, 'AvgOrderValue' élevé	Offrir des offres promotionnelles à ces clients pour leur fidélité
3	'TotalSales' bas, 'OrderCount' médian, 'AvgOrderValue' bas	Amener ces clients à augmenter le volume d'achat en les sollicitant plus (mailing, etc)

- ☐ Top 5 des produits les plus vendu pour le cluster n°2 :
- ☐ Ces informations pourront nous servir afin de cibler mieux nos clients dans le cadre de nos stratégie de campagne Marketing

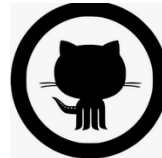
Description__c	SBQQ__ProductName__c
GENA03	2460
GENA09	2297
GENA19	2263
PC1	2200
GENA18	2116

# VI. Conclusion

# Conclusion

- ❑ Cette étude a permis l'utilisation de méthode de classification par la méthode des k-means à partir de 3 variables : **Total des ventes, Nombre de Commandes, Moyenne d'achat des Commandes,**
- ❑ **Nous avons pu segmenter nos clients en 4 segments ou clusters.**
- ❑ **Ceci permet de mieux connaître nos clients et d'optimiser les ventes.**
- ❑ **Au niveau du CRM Salesforce il faut faire apparaître cette appartenance au segment au niveau des écrans clients.**
- ❑ **Une reprise de données sera nécessaire pour cela.**
- ❑ **Enfin, il faudra faire évoluer ce modèle de classification chaque année afin de l'ajuster si nécessaire et de mettre à jour les fiches clients.**

Git





**Merci de votre attention**