

Titre: Customer Segmentation and Purchase Prediction

Resumé

Cette analyse a permis de segmenter efficacement la clientèle à l'aide de l'algorithme de clustering K-means basé sur les métriques RFM (Récence, Fréquence, Valeur Monétaire). Cinq clusters distincts de clients ont été identifiés, chacun présentant des caractéristiques de comportement d'achat spécifiques. L'analyse des moyennes RFM par cluster a révélé des segments clairement définis, allant des clients les plus fidèles et à forte valeur ajoutée (clusters 2 et 4) aux clients réguliers (cluster 0), en passant par les clients à risque (cluster 3) et les clients potentiellement perdus (cluster 1).

Afin d'améliorer la performance d'un modèle de regression logistique et visualiser les données de manière plus concise, une réduction de dimensionnalité a été appliquée à l'aide de l'Analyse en Composantes Principales (ACP). Les deux premières composantes principales ont réussi à capturer une part significative (environ 85.76%) de la variance totale des données RFM, permettant une représentation bidimensionnelle de l'information. L'évaluation d'un modèle de regression logistique entraîné pour prédire l'appartenance à ces clusters a démontré une précision globale élevée (environ 97%). Cependant, l'analyse détaillée par cluster a révélé des performances variables. Les clusters les plus importants en termes de taille (0, 1 et 3) ont été prédits avec une grande précision et un bon rappel. Le cluster 2 a montré une performance correcte, tandis que le cluster 4, de très petite taille, a présenté des difficultés de prédiction. La matrice de confusion a permis de visualiser les types d'erreurs commises par le modèle pour chaque cluster.

1.) Description du jeu de données "Online Retail"

Le jeu de données "Online Retail" est un ensemble de données transactionnelles qui contient les ventes en ligne d'un détaillant britannique non alimentaire. Il couvre la période du 1er décembre 2010 au 9 décembre 2011. Les principales caractéristiques de ce jeu de données sont les suivantes :

- Nombre d'instances : Plus de 500 000
- Nombre d'attributs: 8
 - InvoiceNo : Numéro de facture
 - StockCode : Code du produit
 - Description : Description du produit
 - Quantity : Quantité de produits achetés
 - InvoiceDate : Date et heure de la facture
 - UnitPrice : Prix unitaire du produit

- CustomerID : Identifiant du client
- Country: Pays du client

2.) Chargement, exploration et nettoyage des données

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: # Charger les données depuis un fichier xlsx
file_path ="C:/Users/E ROOT/Desktop/TP_Maths_Apprentissage_Automatique_DSM1/onlydata.xlsx"
data = pd.read_excel(file_path)
```

```
In [3]: # Explorer les données
print(data.head())
print(data.info())
```

	InvoiceNo	StockCode	Description	Quantity	\
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	
1	536365	71053	WHITE METAL LANTERN	6	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	

	InvoiceDate	UnitPrice	CustomerID	Country
0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
Column Non-Null Count Dtype

0 InvoiceNo 541909 non-null object
1 StockCode 541909 non-null object
2 Description 540455 non-null object
3 Quantity 541909 non-null int64
4 InvoiceDate 541909 non-null datetime64[ns]
5 UnitPrice 541909 non-null float64
6 CustomerID 406829 non-null float64
7 Country 541909 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
None

```
In [4]: # Nettoyage
data = data[data['CustomerID'].notnull()]
data = data[data['Quantity'] > 0]
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])
```

```
In [5]: print(data.head())
print(data.info())
```

```

InvoiceNo StockCode          Description  Quantity \
0      536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER      6
1      536365     71053           WHITE METAL LANTERN      6
2      536365    84406B  CREAM CUPID HEARTS COAT HANGER      8
3      536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE      6
4      536365    84029E      RED WOOLLY HOTTIE WHITE HEART.      6

   InvoiceDate  UnitPrice  CustomerID  Country
0 2010-12-01 08:26:00      2.55  17850.0  United Kingdom
1 2010-12-01 08:26:00      3.39  17850.0  United Kingdom
2 2010-12-01 08:26:00      2.75  17850.0  United Kingdom
3 2010-12-01 08:26:00      3.39  17850.0  United Kingdom
4 2010-12-01 08:26:00      3.39  17850.0  United Kingdom
<class 'pandas.core.frame.DataFrame'>
Index: 397924 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
--- 
 0   InvoiceNo     397924 non-null   object 
 1   StockCode     397924 non-null   object 
 2   Description   397924 non-null   object 
 3   Quantity      397924 non-null   int64  
 4   InvoiceDate   397924 non-null   datetime64[ns]
 5   UnitPrice     397924 non-null   float64
 6   CustomerID   397924 non-null   float64
 7   Country       397924 non-null   object 
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 27.3+ MB
None

```

3.) Présentation de la métrique RFM

La métrique RFM est une méthode d'analyse marketing utilisée pour segmenter les clients en fonction de leur comportement d'achat. RFM est un acronyme qui signifie:

- Récence (R) : Mesure le temps écoulé depuis le dernier achat d'un client. Plus la récence est faible, plus le client est considéré comme actif et engagé. Fréquence (F) : Indique le nombre de transactions sur une période donnée. Une fréquence élevée suggère un client fidèle. Montant (M) : Représente la valeur monétaire totale des achats d'un client. Un montant élevé indique un client précieux.

L'analyse RFM permet aux entreprises de mieux comprendre leurs clients et de personnaliser leurs stratégies marketing. Par exemple :

- Les clients avec une récence, une fréquence et un montant élevés sont considérés comme les plus précieux et méritent une attention particulière.
- Les clients avec une récence faible mais une fréquence et un montant faibles peuvent être ciblés avec des offres pour les inciter à augmenter leurs achats.
- Les clients avec une récence élevée sont considérés comme des clients perdus et

une stratégie de réactivation peut être mise en place. En résumé, la métrique RFM est un outil puissant pour la segmentation de la clientèle et l'optimisation des campagnes marketing.

4.) Application de la méthode de clustering K-means pour segmenter les clients

Nous appliquons l'algorithme K-means pour segmenter les clients sur la base des données RFM.

```
In [6]: # Créer la table RFM, il s'agit du data frame qui sera utilisé dans la suite de
now = data['InvoiceDate'].max() + pd.Timedelta(days=1)
rfm = data.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (now - x.max()).days,
    'InvoiceNo': 'nunique',
    'UnitPrice': lambda x: (x * data.loc[x.index, 'Quantity']).sum()
})
rfm.columns = ['Recency', 'Frequency', 'Monetary']
```

```
In [7]: print(rfm.head())
```

	Recency	Frequency	Monetary
CustomerID			
12346.0	326	1	77183.60
12347.0	2	7	4310.00
12348.0	75	4	1797.24
12349.0	19	1	1757.55
12350.0	310	1	334.40

```
In [8]: from sklearn.preprocessing import StandardScaler
# Standardiser les variables de la table RFM
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm)
```

```
In [9]: print(rfm_scaled)
```

```
[[ 2.33485829e+00 -4.24674873e-01  8.35963391e+00]
 [-9.05199314e-01  3.54080191e-01  2.51046358e-01]
 [-1.75186336e-01 -3.52973410e-02 -2.85464504e-02]
 ...
 [-8.45198247e-01 -2.94882363e-01 -2.08712435e-01]
 [-8.85198958e-01  1.52221279e+00  4.57171586e-03]
 [-4.95192025e-01 -1.65089852e-01 -2.40912314e-02]]
```

```
In [10]: from sklearn.cluster import KMeans
# Méthode du coude pour trouver k optimal
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(rfm_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='o')
plt.xlabel('Figure 1: Nombre de clusters')
```

```
plt.ylabel('WCSS')
plt.title('Méthode du coude')
plt.show()
```

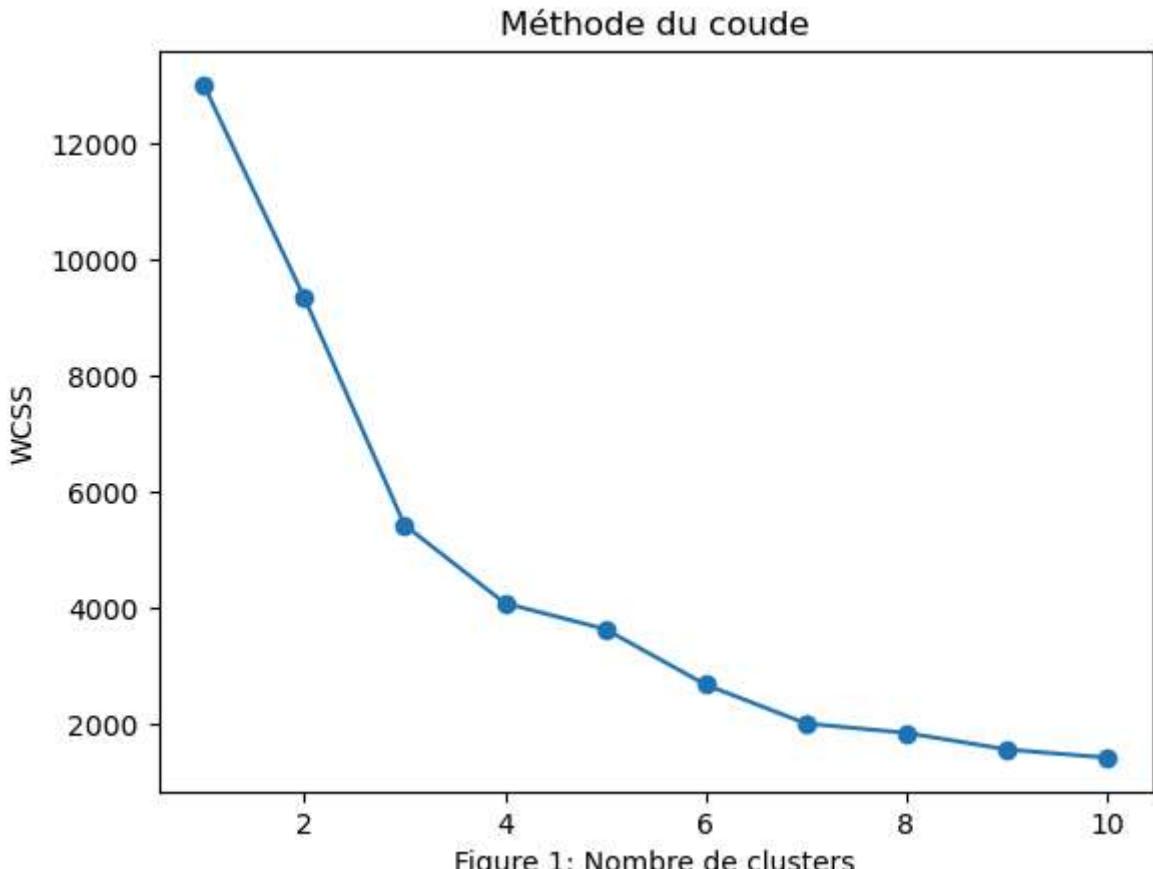


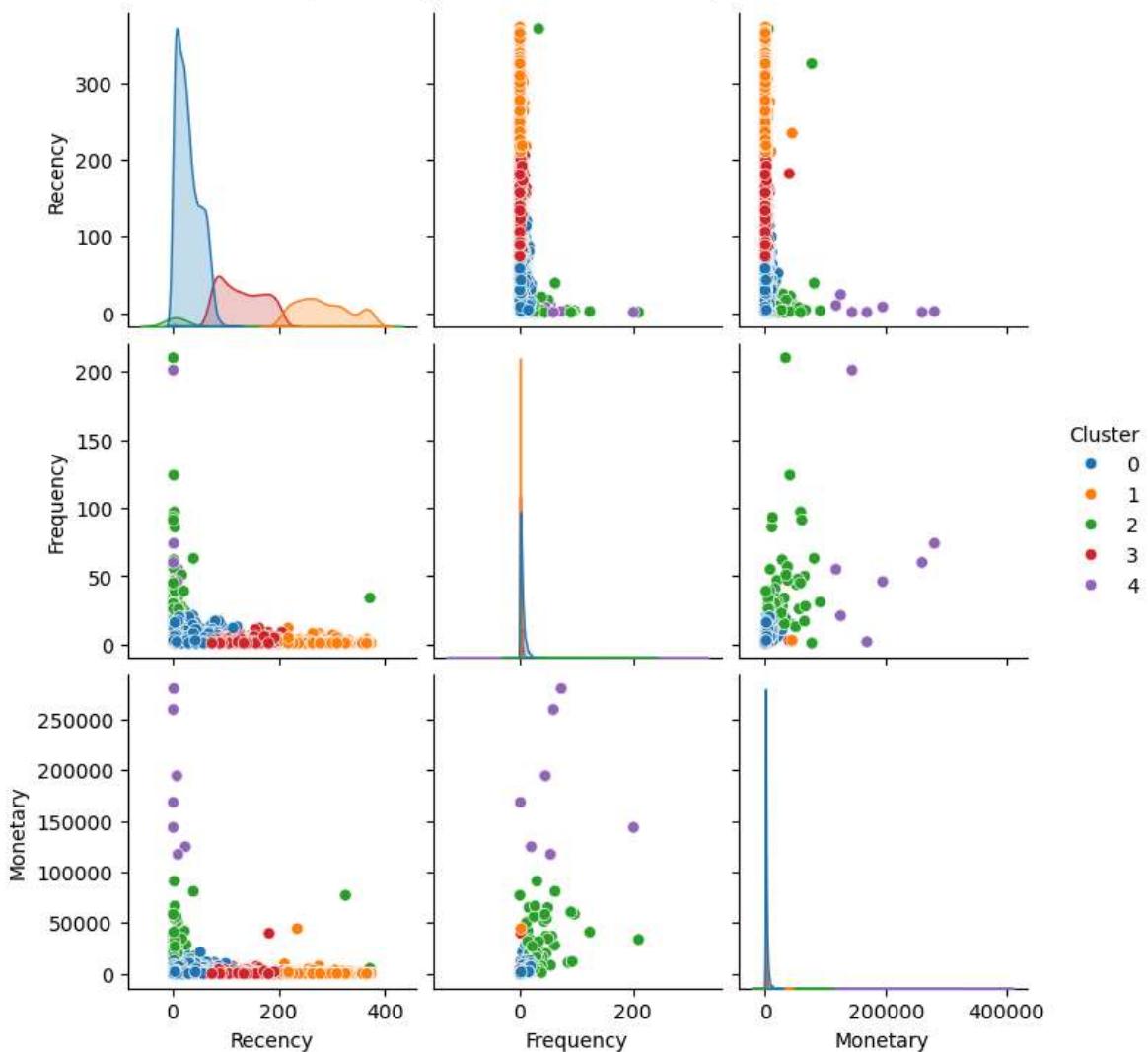
Figure 1: Nombre de clusters

```
In [12]: # Appliquer K-Means avec k=5
kmeans = KMeans(n_clusters=5, random_state=42)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)
print(rfm.head())
```

CustomerID	Recency	Frequency	Monetary	Cluster
12346.0	326	1	77183.60	2
12347.0	2	7	4310.00	0
12348.0	75	4	1797.24	0
12349.0	19	1	1757.55	0
12350.0	310	1	334.40	1

```
In [13]: # Visualisation
sns.pairplot(rfm, hue='Cluster', palette='tab10', diag_kind='kde')
plt.suptitle('Figure 2: Segmentation des clients par K-Means', y=1.02)
plt.show()
```

Figure 2: Segmentation des clients par K-Means



```
In [33]: # Affichage des stats par cluster
print("\n TABLEAU 1:Valeurs des centroides")
print(rfm.groupby('Cluster').mean().round(1))
```

Cluster	TABLEAU 1:Valeurs des centroides		
	Recency	Frequency	Monetary
0	28.8	4.6	1745.5
1	284.6	1.4	501.7
2	13.5	35.4	22165.7
3	129.8	2.1	765.9
4	6.7	65.6	184143.7

4.) Application de l'ACP pour réduire la dimensionnalité

Nous appliquons PCA pour réduire la dimensionnalité des données RFM à 2 composantes principales (PCA1 et PCA2). Cela facilite la visualisation et peut améliorer les performances du modèle de régression logistique.

```
In [15]: from sklearn.decomposition import PCA
# Réduction de dimensionnalité avec PCA
```

```
pca = PCA(n_components=2)
rfm_pca = pca.fit_transform(rfm_scaled)
```

In [16]:

```
# Créer un dataframe avec les composantes principales
pca_df = pd.DataFrame(data=rfm_pca, columns=['PC1', 'PC2'])
# Affichage de la variance expliquée par chaque composante
variance_expliquee=pca.explained_variance_ratio_
print("Variance expliquée par chaque composante:",variance_expliquee)
# Affichage des premières lignes du dataframe obtenu avec les composantes principales
print("Premières lignes de pca_df")
print(pca_df.head())
```

Variance expliquée par chaque composante: [0.55508353 0.30254725]

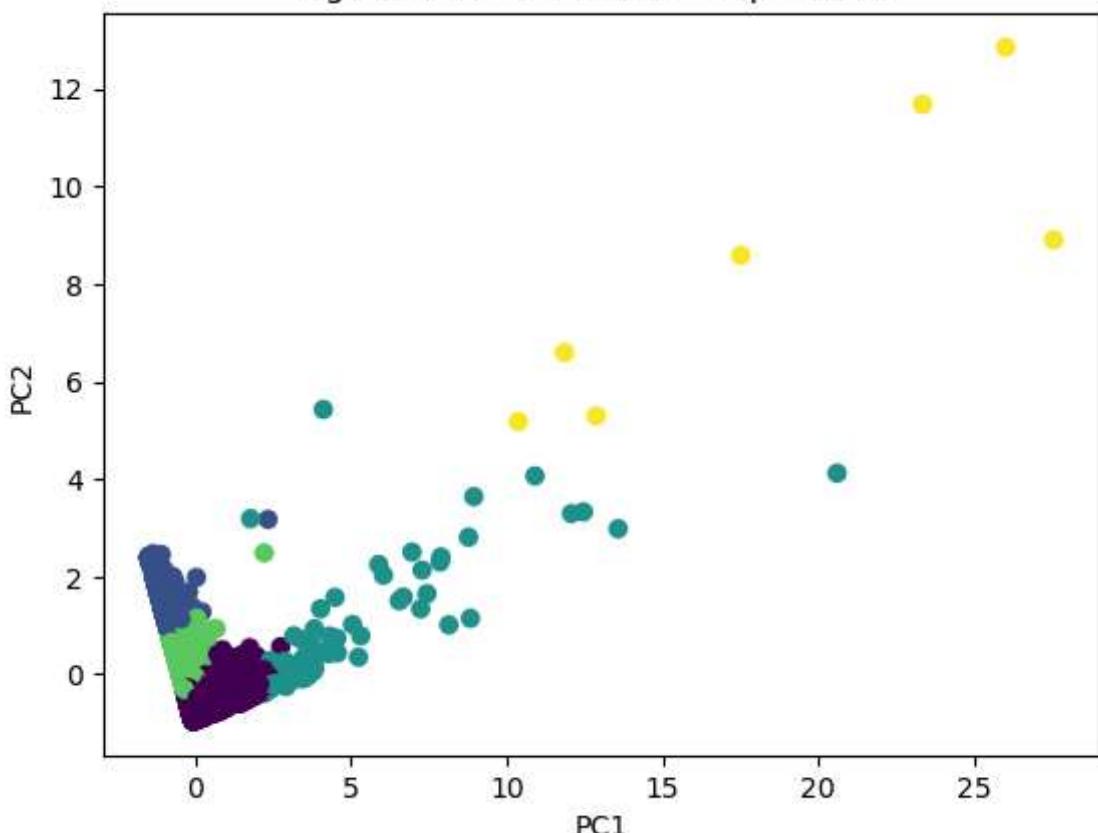
Premières lignes de pca_df

	PC1	PC2
0	4.110109	5.429102
1	0.741861	-0.671587
2	0.024714	-0.174807
3	-0.028120	-0.735154
4	-1.234394	1.835991

In [17]:

```
# Visualiser les clusters dans l'espace des composantes principales
plt.scatter(pca_df['PC1'], pca_df['PC2'], c=rfm['Cluster'])
plt.title('Figure 3: Clusters dans l\'espace ACP')
plt.xlabel('PC1')
plt.ylabel('PC2')
# plt.legend()
plt.show()
```

Figure 3: Clusters dans l'espace ACP



5.) Régression logistique

Nous utilisons la régression logistique pour prédire le cluster d'appartenance d'un client en fonction de ses composantes PCA1 et PCA2. Étant donné que nous avons 5 clusters, il s'agit d'un problème de classification multiclasse. Après cela nous évaluons le modèle en utilisant la précision, le rapport de classification et la matrice de confusion.

```
In [18]: # Préparer les données pour la régression logistique
X = pca_df
y = rfm['Cluster'] # On utilise les clusters comme modalité de la variable cible
```

```
In [19]: from sklearn.model_selection import train_test_split

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_
```

```
In [20]: from sklearn.linear_model import LogisticRegression

# Entrainer le modèle de régression logistique
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
Out[20]: ▾ LogisticRegression ⓘ ?
```

```
LogisticRegression()
```

```
In [21]: # Faire des prédictions sur l'ensemble de test
predictions = model.predict(X_test)
probabilite_positive = model.predict_proba(X_test)[:,1] # Probabilités de la classe
```

```
In [32]: import sklearn.metrics as sm

# Évaluer la précision du modèle
accuracy = sm.accuracy_score(y_test, predictions)
print(f'---Précision du modèle : {accuracy}')
print("\n TABLEAU 2: Rapport de classification")
print(sm.classification_report(y_test, predictions))
#print("\n--- Matrice de confusion -----")

#print(cm)
```

---Précision du modèle : 0.9769585253456221

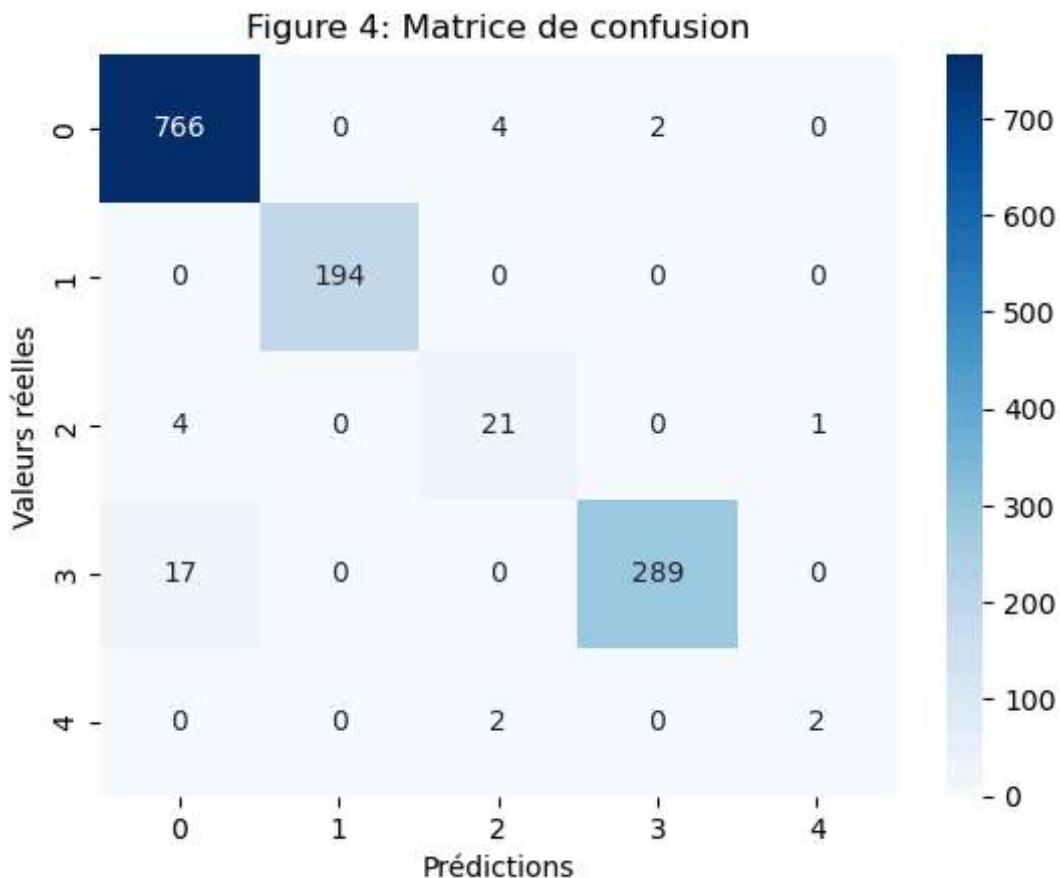
TABLEAU 2: Rapport de classification

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.97	0.99	0.98	772
1	1.00	1.00	1.00	194
2	0.78	0.81	0.79	26
3	0.99	0.94	0.97	306
4	0.67	0.50	0.57	4

accuracy			0.98	1302
macro avg	0.88	0.85	0.86	1302
weighted avg	0.98	0.98	0.98	1302

```
In [26]: # Visualisation de la matrice de confusion
cm=sm.confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Prédictions")
plt.ylabel("Valeurs réelles")
plt.title("Figure 4: Matrice de confusion")
plt.show()
```



```
In [51]: # Probabilités prédictes pour La classe positive
print("Probabilités prédictes (classe 1):",probabilite_positive)
```

Probabilités prédictes (classe 1): [1.34545110e-06 9.31863486e-06 1.52639511e-09
... 2.72026537e-04
6.63991423e-11 1.41609049e-08]

6.) Résultats et discussion

K-mean

- Dans le graphique (figure 1), le WCSS chute fortement de 1 à environ 4 clusters. Après 4 clusters, la diminution du WCSS devient beaucoup moins prononcée, créant ce qui ressemble à un coude autour de 4 clusters.

Par conséquent, sur la base de la méthode du coude, un nombre optimal potentiel de clusters serait de 4. Bien qu'un nombre de clusters dépassant 4 ne semblerait pas apporter pas d'information consistante, nous avons poursuivi l'analyse avec 5 clusters pour respecter la consigne.

- Interprétation détaillée de chaque cluster (voir Tableau 1) :
 - Cluster 0 : Composé de clients avec une récence relativement bonne (28.8 jours), une fréquence d'achat modérée (4.6 transactions) et une valeur monétaire correcte (1745.5). Ce sont des clients assez récents qui achètent un nombre raisonnable de fois et dépensent un montant décent.
 - Cluster 1 : Composé de clients avec une récence très élevée (284.6 jours), une fréquence d'achat très faible (1.4 transaction) et une valeur monétaire faible (501.7). Ce sont des clients qui n'ont pas acheté depuis longtemps, achètent rarement et dépensent peu. Ils sont probablement des clients inactifs ou perdus.
 - Cluster 2 : Composé de clients avec une récence très faible (13.5 jours), une fréquence d'achat très élevée (35.4 transactions) et une valeur monétaire élevée (22165.7). Ce sont des clients récents, très fréquents et qui dépensent beaucoup. Ce sont vos clients les plus fidèles et les plus précieux.
 - Cluster 3 : Composé de clients avec une récence moyenne (129.8 jours), une fréquence d'achat faible (2.1 transactions) et une valeur monétaire modérée (765.9). Ce sont des clients qui ont acheté il y a un certain temps, n'achètent pas souvent et leur dépense moyenne n'est pas très élevée. Ils pourraient être des clients à risque de perte.
 - Cluster 4 : Composé de clients avec une récence extrêmement faible (6.7 jours), une fréquence d'achat exceptionnellement élevée (65.6 transactions) et une valeur monétaire très élevée (184143.7). Ce sont vos clients les plus récents, les plus fréquents et ceux qui dépensent le plus, surpassant même le cluster 2 en termes de fréquence et de dépenses totales. Ce sont vos clients les plus précieux.

PAC

Nous avons créé un nouveau DataFrame appelé `pca_df` où les colonnes représentent les deux composantes principales que la PCA a identifiées : PC1 (Première Composante Principale) et PC2 (Deuxième Composante Principale). Les valeurs dans ces colonnes sont les coordonnées de chaque point de données RFM projeté sur ces deux nouvelles composantes.

- La Première Composante Principale (PC1) explique environ 55.51% de la variance totale des données RFM. C'est la direction dans les données qui capture la plus grande partie de la dispersion des points.
- La Deuxième Composante Principale (PC2) explique environ 30.25% de la variance totale des données RFM. C'est la deuxième direction la plus importante en termes de variance, et elle est orthogonale (non corrélée) à la PC1.

Ensemble, les deux premières composantes principales expliquent environ 85.76% de la variance totale des données RFM originales. Cela suggère que ces deux dimensions

résument une part considérable de l'information contenue dans les trois dimensions RFM initiales.

Le DataFrame pca_df contient les données RFM transformées dans cet espace réduit, qui nous avons ensuite visualisé (figure 3) d'une part et d'autre part elles ont été utilisées entrées pour le modèle de régression logistique.

Regression logistique

Nous avons entraîné un modèle de régression logistique avec 70% des données du DataFrame pca_df, pour prédire l'appartenance aux clusters K-means et nous avons ensuite évalué sa performance avec les 30% restantes.

Voici l'interprétation des résultats que vous avez obtenus :

- Précision globale du modèle: Elle indique le pourcentage de toutes les prédictions faites par le modèle qui étaient correctes. Dans ce cas, environ 97.70% des prédictions d'appartenance aux clusters étaient justes. C'est une très bonne performance globale.
- Rapport de classification :Analyse par cluster :
 - Cluster 0 : Très bonne précision (0.97) et excellent rappel (0.99), résultant en un score F1 élevé (0.98). Le modèle est très performant pour ce cluster, qui est aussi le plus représenté.
 - Cluster 1 : Précision et rappel parfaits (1.00), donc un score F1 de 1.00. Le modèle identifie parfaitement toutes les instances de ce cluster et ne fait pas de fausses prédictions pour celui-ci.
 - Cluster 2 : Précision de 0.78 et rappel de 0.81, avec un score F1 de 0.79. La performance est bonne, mais un peu moins élevée que pour les clusters 0 et 1. Il y a quelques erreurs de classification pour ce cluster.
 - Cluster 3 : Très bonne précision (0.99) et bon rappel (0.94), avec un score F1 de 0.97. Le modèle est performant pour ce cluster également.
 - Cluster 4 : Précision faible (0.67) et rappel très faible (0.50), résultant en un score F1 faible (0.57). La performance est nettement moins bonne pour ce cluster, qui est aussi extrêmement petit (seulement 4 instances dans l'ensemble de test). Le modèle a du mal à identifier correctement les instances de ce cluster et fait plus d'erreurs.
- Mesures agrégées :
 - accuracy (exactitude) : Confirmée à 0.98, ce qui correspond à la précision globale mentionnée précédemment.
- Matrice de confusion :
 - Pour le cluster 0, 766 instances ont été correctement prédites, 0 ont été incorrectement prédites comme étant dans d'autres clusters. 4 instances du cluster 0 ont été mal classées comme cluster 2, et 2 comme cluster 3.
 - Pour le cluster 1, les 194 instances ont été correctement prédites.

- Pour le cluster 2, 21 instances ont été correctement prédites. 4 instances du cluster 2 ont été mal classées comme cluster 0, et 1 comme cluster 3.
- Pour le cluster 3, les 289 instances ont été correctement prédites.
- Pour le cluster 4, seulement 2 instances ont été correctement prédites. 2 instances du cluster 4 ont été mal classées comme cluster 2.

La matrice de confusion confirme les observations faites à partir du rapport de classification : excellente performance pour les clusters 0, 1 et 3, performance correcte pour le cluster 2, et performance faible pour le cluster 4 (probablement en raison de sa très petite taille).

Le modèle de régression logistique que vous avez entraîné pour prédire l'appartenance aux clusters K-means est globalement très précis (97%). Cependant, il est important de noter que la performance varie considérablement d'un cluster à l'autre, avec des difficultés notables pour le cluster 4. Néanmoins selon ce modèle la segmentation des clients est pertinente et être utilisée pour cibler des campagnes marketing.

Conclusion

En conclusion, cette analyse RFM, combinée au clustering K-means et à l'évaluation d'un modèle de régression logistique, fournit des informations précieuses pour une segmentation de la clientèle et une compréhension approfondie de leurs comportements d'achat. Ces insights peuvent être directement exploités pour élaborer des stratégies marketing ciblées et personnalisées pour chaque segment, dans le but d'optimiser l'engagement client, la fidélisation et, ultimement, la valeur client globale. Il est important de tenir compte du déséquilibre des classes, notamment pour le cluster 4, lors de la mise en œuvre de stratégies basées sur ces prédictions. Des efforts supplémentaires pourraient être envisagés pour améliorer la prédiction des clusters minoritaires si cela s'avère crucial pour les objectifs commerciaux.