Churn client

December 19, 2025

1 *CONTEXTE*

Une entreprise de télécommunications souhaite anticiper le départ de ses clients (churn) pour mettre en place des actions de rétention ciblées. L'objectif est de construire un modèle prédictif capable d'identifier les clients à risque de résiliation dans les 30 prochains jours, afin de leur proposer des offres personnalisées avant qu'ils ne partent.

2 *Objectifs*

- Identifier les facteurs principaux qui influencent le départ des clients
- Prédire la probabilité de churn pour chaque client
- Proposer un score de risque pour prioriser les actions marketing

3 *Source de données*

https://www.kaggle.com/datasets/blastchar/telco-customer-churn

4 *Description du dataset*

4.1 *Général*

7 043 lignes (clients) et 21 colonnes incluant :

- Informations démographiques (genre, âge, personnes à charge)
- Services souscrits (téléphone, internet, streaming, sécurité)
- Informations contractuelles (type de contrat, mode de paiement)
- Montants facturés (mensuel et total)
- Variable cible : Churn (Yes/No)

4.2 *Détail*

- customerID : Identifiant du client
- gender : Indique si le client est un homme ou une femme
- SeniorCitizen : Indique si le client est une personne âgée ou non (1, 0)
- Partner : Indique si le client a un partenaire ou non (Oui, Non)
- Dependents : Indique si le client a des personnes à charge ou non (Oui, Non)
- tenure : Nombre de mois pendant lesquels le client est resté avec l'entreprise
- PhoneService : Indique si le client dispose d'un service téléphonique ou non (Oui, Non)

- MultipleLines : Indique si le client a plusieurs lignes ou non (Oui, Non, Pas de service téléphonique)
- InternetService : Fournisseur de service Internet du client (DSL, Fibre optique, Aucun)
- OnlineSecurity : Indique si le client dispose d'une sécurité en ligne ou non (Oui, Non, Pas de service Internet)
- OnlineBackup : Indique si le client dispose d'une sauvegarde en ligne ou non (Oui, Non, Pas de service Internet)
- DeviceProtection : Indique si le client dispose d'une protection d'appareil ou non (Oui, Non, Pas de service Internet)
- TechSupport : Indique si le client dispose d'une assistance technique ou non (Oui, Non, Pas de service Internet)
- StreamingTV : Indique si le client dispose d'un service de TV en streaming ou non (Oui, Non, Pas de service Internet)
- StreamingMovies : Indique si le client dispose d'un service de films en streaming ou non (Oui, Non, Pas de service Internet)
- Contract : Type de contrat du client (Mensuel, Un an, Deux ans)
- PaperlessBilling : Indique si le client utilise la facturation électronique ou non (Oui, Non)
- PaymentMethod : Méthode de paiement du client (Chèque électronique, Chèque envoyé, Virement bancaire automatique, Carte de crédit automatique)
- MonthlyCharges : Montant facturé au client chaque mois
- TotalCharges : Montant total facturé au client
- Churn : Indique si le client s'est désabonné ou non (Oui, Non)

5 *RESULTATS*

L'analyse a permis de développer un système de prédiction capable d'identifier à l'avance les clients qui risquent de quitter l'entreprise. Après avoir testé et comparé plusieurs approches différentes, la solution la plus efficace détecte correctement quatre clients sur cinq qui vont effectivement partir, soit 298 clients identifiés sur les 374 qui résiliront réellement leur abonnement dans les données analysées.

Cette performance représente une amélioration considérable par rapport aux premières versions du système qui ne détectaient qu'un client sur deux. Concrètement, le nombre de clients à risque qui passaient inaperçus est passé de 180 à seulement 76, soit une réduction de plus de la moitié. Ces clients manqués représentent un véritable coût pour l'entreprise puisqu'ils partiront sans avoir bénéficié d'aucune action pour les retenir. Le système fait parfois des erreurs en identifiant comme étant à risque des clients qui resteront finalement fidèles, générant ainsi des alertes inutiles. Cependant, ce compromis a été volontairement accepté car l'objectif prioritaire est de ne laisser échapper aucun client réellement à risque. En effet, proposer une offre de rétention à un client fidèle coûte peu à l'entreprise, alors que perdre un client qui n'a pas été détecté représente une perte de revenus bien plus importante sur le long terme.

L'analyse du système de prédiction révèle de manière très claire quels sont les facteurs qui influencent le plus la décision des clients de partir ou de rester.

- Le type d'engagement contractuel domine absolument tous les autres éléments en représentant à lui seul plus de 60% du pouvoir de prédiction du système. Les chiffres sont éloquents : parmi les clients sans engagement (contrat mensuel), 43% finissent par partir, alors que chez ceux engagés sur deux ans, seulement 3% quittent l'entreprise. Le risque de départ est donc

quatorze fois plus élevé pour un client sans engagement, faisant de la durée du contrat le levier de rétention de loin le plus puissant dont dispose l'entreprise.

- Le type de service internet souscrit constitue le deuxième facteur en importance, pesant environ 12% dans la décision de prédiction. Ce qui est particulièrement préoccupant, c'est que l'offre fibre optique, censée être le service premium de l'entreprise, présente un taux de départ catastrophique de 42%. Cela signifie que plus de deux clients sur cinq qui ont choisi cette offre haut de gamme finissent par résilier, suggérant soit des problèmes de qualité de service, soit un prix perçu comme trop élevé par rapport à la valeur reçue, soit une concurrence particulièrement agressive sur ce segment.
- L'analyse a également permis de créer un indicateur de profil à très haut risque qui combine trois caractéristiques dangereuses : un contrat mensuel, une ancienneté de moins d'un an et un paiement par chèque électronique. Cet indicateur synthétique s'est révélé très pertinent puisqu'il se classe en quatrième position parmi tous les facteurs pris en compte par le système. Les clients correspondant à ce profil présentent un taux de départ de 63%, soit près de deux sur trois, alors que les autres clients ne partent qu'à 21%. Cette identification d'un segment ultra-risque facilite grandement le ciblage des actions marketing les plus intensives.
- Les services additionnels souscrits par les clients jouent également un rôle protecteur significatif. Les clients qui ont ajouté des options comme le streaming de films ou de télévision à leur abonnement de base ont tendance à rester plus fidèles, ces services représentant chacun environ 3% du pouvoir de prédiction. Cela confirme que la richesse et la diversité de l'offre créent de la valeur perçue et renforcent l'attachement du client à l'opérateur. À l'inverse, les clients qui se contentent de l'offre de base minimale sont beaucoup plus volatiles.
- Les facteurs comme l'ancienneté du client et le montant de sa facture mensuelle, bien qu'importants, jouent un rôle moins décisif qu'on aurait pu l'imaginer. L'ancienneté compte pour environ 3% et le montant facturé pour moins de 1% dans les prédictions du système. Cette importance relative plus faible suggère que ces aspects sont en grande partie déjà capturés et synthétisés par les autres facteurs plus complexes que le système prend en compte.

Au-delà de sa capacité à prédire correctement les départs, le système développé constitue un véritable outil de travail directement utilisable par les équipes commerciales et marketing. Chaque client de la base se voit attribuer un score de risque exprimé en pourcentage de zéro à cent, permettant de mesurer précisément le niveau de danger et d'adapter finement les actions à mener. Cette granularité fine offre une grande flexibilité dans la personnalisation des offres de rétention.

Pour faciliter l'utilisation opérationnelle, ces scores continus ont été regroupés en quatre catégories de risque bien distinctes. Les clients en situation CRITIQUE, avec un score supérieur à 70%, nécessitent une intervention humaine immédiate dans les quarante-huit heures. Pour ces clients, l'entreprise doit mobiliser ses meilleurs conseillers et proposer des remises importantes pouvant aller jusqu'à la moitié de la facture pendant plusieurs mois, ainsi que des passages gratuits vers des contrats plus longs. L'urgence absolue s'impose car ces clients sont sur le point de partir.

Les clients classés à risque ÉLEVÉ, avec un score entre 50 et 70%, justifient un contact proactif dans la semaine qui suit l'identification. Les offres proposées peuvent être un peu moins agressives, incluant des réductions modérées sur quelques mois et des services additionnels gratuits comme le support technique ou la sécurité en ligne. L'objectif est de renouer le dialogue avant que la situation ne se dégrade davantage et de démontrer au client qu'on se soucie de lui.

Pour les clients en risque MODÉRÉ, avec un score entre 30 et 50%, des actions moins intrusives mais néanmoins attentives sont recommandées. Des campagnes par email personnalisées, des enquêtes de satisfaction pour identifier d'éventuels problèmes naissants, et des offres de services complémentaires

constituent une approche adaptée. Il s'agit davantage de prévention que d'intervention d'urgence.

Enfin, les clients à risque FAIBLE, avec un score inférieur à 30%, ne requièrent qu'un suivi de routine avec les programmes de fidélité standards de l'entreprise. Ces clients satisfaits et fidèles peuvent être contactés pour des offres d'amélioration de leur forfait mais sans pression particulière, l'objectif étant simplement de maintenir leur satisfaction élevée.

Ce système de catégorisation a été appliqué à l'ensemble des sept mille clients de la base, créant ainsi une liste complète, priorisée et immédiatement actionnable. Les équipes commerciales savent exactement qui contacter en premier, avec quel niveau d'urgence, et quel type d'offre proposer. Cette clarté opérationnelle transforme les prédictions statistiques en actions business concrètes et mesurables.

6 *CODE*

6.1 *Importation des librairies*

[1219]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
```

6.2 *Etape 1 : Chargement des données*

[1221]:
```
data_initial = pd.read_csv(r'C:\Users\inhou\Documents\PROJETS\Python\CHURN␣
 ↪CLIENT TELECOM\Telco-Customer-Churn.csv')
```

[1222]:
```
# Dimensions
data_initial.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
```

```
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[1223]:
```
# Extrait
data_initial.head(5)
```

[1223]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	…	No

	MultipleLines	InternetService	OnlineSecurity	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	No phone service	DSL	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	No	DSL	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	No	DSL	Yes	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	No phone service	DSL	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	No	Fiber optic	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

```
[5 rows x 21 columns]
```

[1224]:
```
# Conversion en valeurs numériques
data_initial['TotalCharges'] = pd.to_numeric(data_initial['TotalCharges'],␣
 ↪errors='coerce')
```

[1225]:
```
# Statisiques descriptives de bases
data_initial.describe()
```

[1225]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	32.371149	64.761692	2283.300441
std	0.368612	24.559481	30.090047	2266.771362
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.500000	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.850000	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

[1226]:
```
data_initial.describe(include=['object'])
```

[1226]:

	customerID	gender	Partner	Dependents	PhoneService	MultipleLines
count	7043	7043	7043	7043	7043	7043
unique	7043	2	2	2	2	3
top	3186-AJIEK	Male	No	No	Yes	No
freq	1	3555	3641	4933	6361	3390

	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
count	7043	7043	7043	7043
unique	3	3	3	3
top	Fiber optic	No	No	No
freq	3096	3498	3088	3095

	TechSupport	StreamingTV	StreamingMovies	Contract
count	7043	7043	7043	7043
unique	3	3	3	3
top	No	No	No	Month-to-month
freq	3473	2810	2785	3875

	PaperlessBilling	PaymentMethod	Churn
count	7043	7043	7043
unique	2	4	2
top	Yes	Electronic check	No
freq	4171	2365	5174

6.3 *Etape 2 : Nettoyage*

[1228]:
```
data_clean = data_initial.copy()
```

6.3.1 *Qualité des données*

[1230]:
```
# Valeurs manquantes
data_clean.isnull().sum()
```

[1230]:
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

11 valeurs manquantes dans TotalCharges. Probablement des nouveaux clients (tenure = 0) qui n'ont pas encore été facturés

[1232]:
```
# Doublons
data_clean.duplicated().sum()
```

[1232]:
```
np.int64(0)
```

Dataset propre. Pas de lignes dupliquée.

6.3.2 *Gestion des données manquantes*

[1235]:
```
# Identification des données manquantes
data_missed = data_clean[data_clean['TotalCharges'].isnull()]
```

[1236]:
```
data_missed
```

[1236]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	\
488	4472-LVYGI	Female	0	Yes	Yes	0	
753	3115-CZMZD	Male	0	No	Yes	0	
936	5709-LVOEQ	Female	0	Yes	Yes	0	
1082	4367-NUYAO	Male	0	Yes	Yes	0	
1340	1371-DWPAZ	Female	0	Yes	Yes	0	
3331	7644-OMVMY	Male	0	Yes	Yes	0	
3826	3213-VVOLG	Male	0	Yes	Yes	0	
4380	2520-SGTTA	Female	0	Yes	Yes	0	
5218	2923-ARZLG	Male	0	Yes	Yes	0	
6670	4075-WKNIU	Female	0	Yes	Yes	0	
6754	2775-SEFEE	Male	0	No	Yes	0	

	PhoneService	MultipleLines	InternetService	OnlineSecurity	…	\
488	No	No phone service	DSL	Yes	…	
753	Yes	No	No	No internet service	…	
936	Yes	No	DSL	Yes	…	
1082	Yes	Yes	No	No internet service	…	
1340	No	No phone service	DSL	Yes	…	
3331	Yes	No	No	No internet service	…	
3826	Yes	Yes	No	No internet service	…	
4380	Yes	No	No	No internet service	…	
5218	Yes	No	No	No internet service	…	
6670	Yes	Yes	DSL	No	…	
6754	Yes	Yes	DSL	Yes	…	

	DeviceProtection	TechSupport	StreamingTV	\
488	Yes	Yes	Yes	
753	No internet service	No internet service	No internet service	
936	Yes	No	Yes	
1082	No internet service	No internet service	No internet service	
1340	Yes	Yes	Yes	
3331	No internet service	No internet service	No internet service	
3826	No internet service	No internet service	No internet service	
4380	No internet service	No internet service	No internet service	
5218	No internet service	No internet service	No internet service	
6670	Yes	Yes	Yes	
6754	No	Yes	No	

	StreamingMovies	Contract	PaperlessBilling	\
488	No	Two year	Yes	
753	No internet service	Two year	No	
936	Yes	Two year	No	
1082	No internet service	Two year	No	
1340	No	Two year	No	
3331	No internet service	Two year	No	
3826	No internet service	Two year	No	

```
4380  No internet service  Two year              No
5218  No internet service  One year             Yes
6670                   No  Two year              No
6754                   No  Two year             Yes

                 PaymentMethod MonthlyCharges  TotalCharges  Churn
488   Bank transfer (automatic)          52.55           NaN     No
753                Mailed check          20.25           NaN     No
936                Mailed check          80.85           NaN     No
1082               Mailed check          25.75           NaN     No
1340    Credit card (automatic)          56.05           NaN     No
3331               Mailed check          19.85           NaN     No
3826               Mailed check          25.35           NaN     No
4380               Mailed check          20.00           NaN     No
5218               Mailed check          19.70           NaN     No
6670               Mailed check          73.35           NaN     No
6754  Bank transfer (automatic)          61.90           NaN     No

[11 rows x 21 columns]
```

On remarque que les clients qui n'ont pas de TotalCharges sont des nouveaux clients donc qui viennent tout juste de s'abonner (tenure = 0)! Le client n'a pas encore complété 1 mois d'abonnement. Il n'a donc pas encore été facturé.

[1238]:
```
# Imputation des valeurs manquantes
data_clean.loc[data_clean['TotalCharges'].isnull(), 'TotalCharges'] = 0
```

[1239]:
```
# Vérification
data_clean.isnull().sum()
```

[1239]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
```

```
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

6.3.3 *Compréhension des variables*

[1241]:
```
# Type de variables
data_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

[1242]:
```
# Modalité des variables catégorielles
for col in data_clean.select_dtypes(include = 'object').columns:
    print(data_clean[col].unique())
```

```
['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' … '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']
['Female' 'Male']
['Yes' 'No']
['No' 'Yes']
```

```
['No' 'Yes']
['No phone service' 'No' 'Yes']
['DSL' 'Fiber optic' 'No']
['No' 'Yes' 'No internet service']
['Yes' 'No' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['Month-to-month' 'One year' 'Two year']
['Yes' 'No']
['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
['No' 'Yes']
```

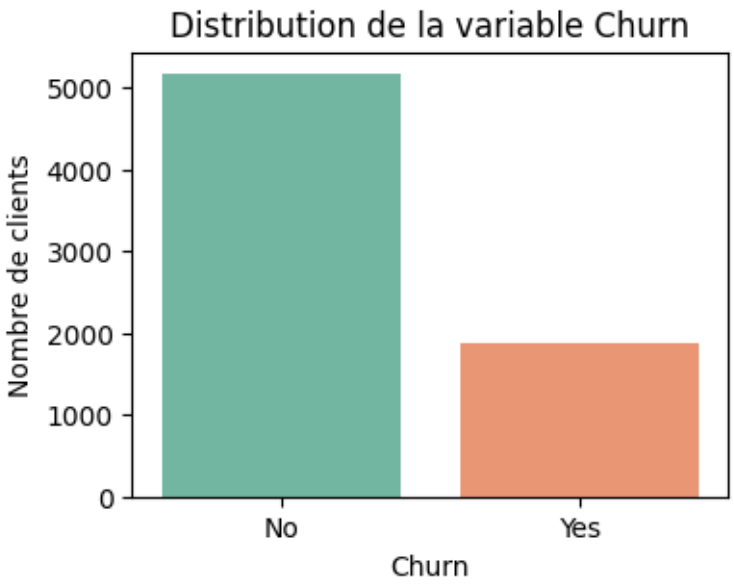
6.4 *Etape 3 : Analyse exploratoire approfondie*

6.4.1 *Analyse univariée*

[1245]:
```
# Distribution du Churn
data_clean['Churn'].value_counts(normalize=True) * 100
```

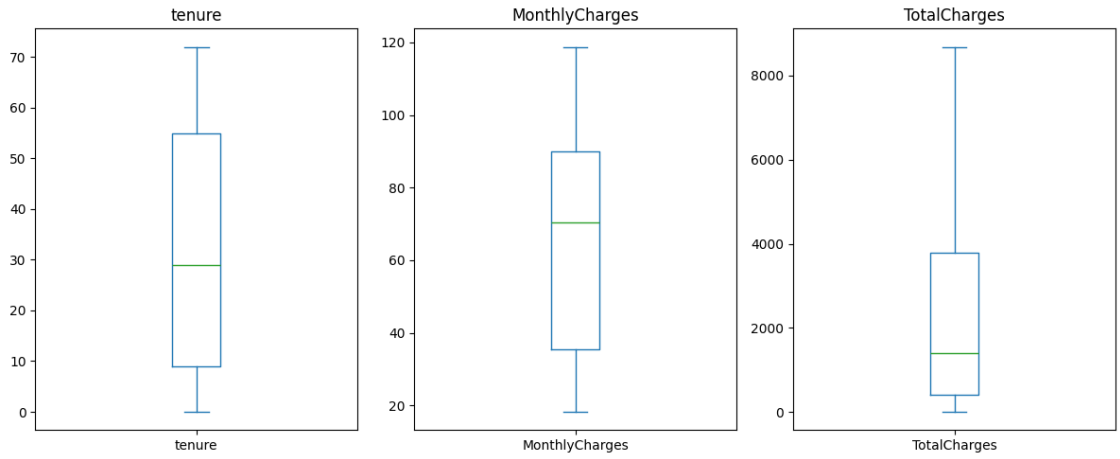
[1245]:
```
Churn
No     73.463013
Yes    26.536987
Name: proportion, dtype: float64
```

[1246]:
```
plt.figure(figsize=(4, 3))
sns.countplot(data=data_clean, x='Churn', hue='Churn', palette='Set2',␣
 ↪legend=False)
plt.title('Distribution de la variable Churn')
plt.ylabel('Nombre de clients')
plt.show()
```



1 client sur 4 quitte l'entreprise ==> le taux de churn est important

[1248]:
```
# Distribution variables numériques
cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
plt.figure(figsize=(12,5))
for i, col in enumerate (cols, 1) :
    plt.subplot(1, 3, i)
    data_clean[col].plot(kind='box')
    plt.title(col)
plt.tight_layout()
plt.show()
```



1. Tenure :

On a une distribution assez uniforme avec une légère concentration vers les nouveaux clients. Beaucoup de clients sont récents (0-10 mois) Clients fidèles (60-72 mois) → probablement moins de churn

2. MonthlyCharges :

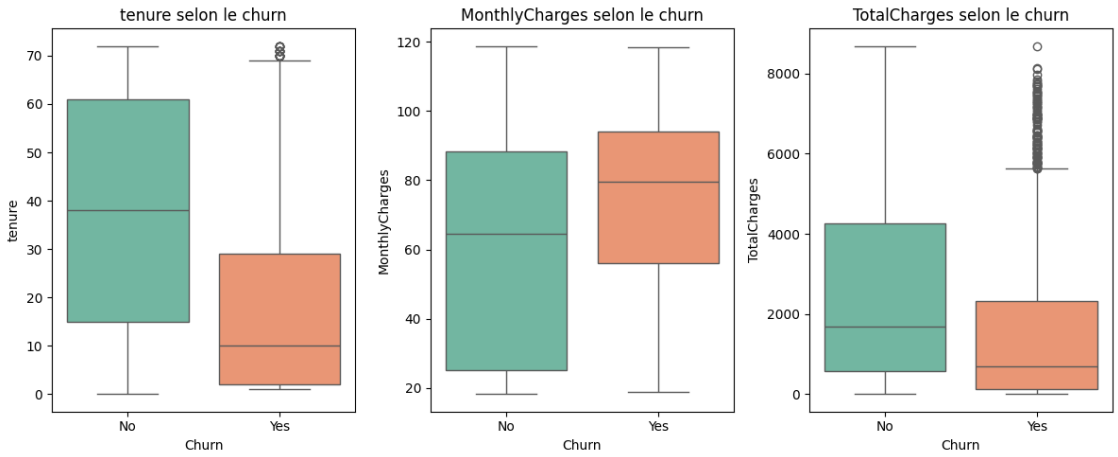
La distribution est bimodale : des petites factures ~20-40$ et des factures élevées ~80-100$. Les factures élevées peuvent être liées au churn (clients insatisfaits du prix ?)

3. TotalCharges :

On note une forte concentration vers les valeurs basses (nouveaux clients) et quelques clients avec des montants très élevés (anciens clients fidèles).

6.4.2 *Analyse bivariée*

[1251]:
```
# Distribution variables numériques et churn
cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
plt.figure(figsize=(12,5))
for i, col in enumerate (cols, 1) :
    plt.subplot(1, 3, i)
    sns.boxplot(data=data_clean, x='Churn', y=col, hue = 'Churn',␣
 ↪palette='Set2', legend=False)
    plt.xlabel('Churn')
    plt.ylabel(col)
    plt.title(f'{col} selon le churn')
plt.tight_layout()
plt.show()
```



Graphique 1 : Tenure selon le Churn

- Clients fidèles (No churn) : médiane ~ 38 mois d'ancienneté
- Clients partis (Yes churn) : médiane ~ 10 mois seulement

Les 12 premiers mois sont critiques pour la rétention : plus un client reste longtemps, plus il devient fidèle.

Graphique 2 : MonthlyCharges selon le churn

- Clients fidèles (No churn) : médiane ~ 65
- Clients partis (Yes churn) : médiane ~ 80$

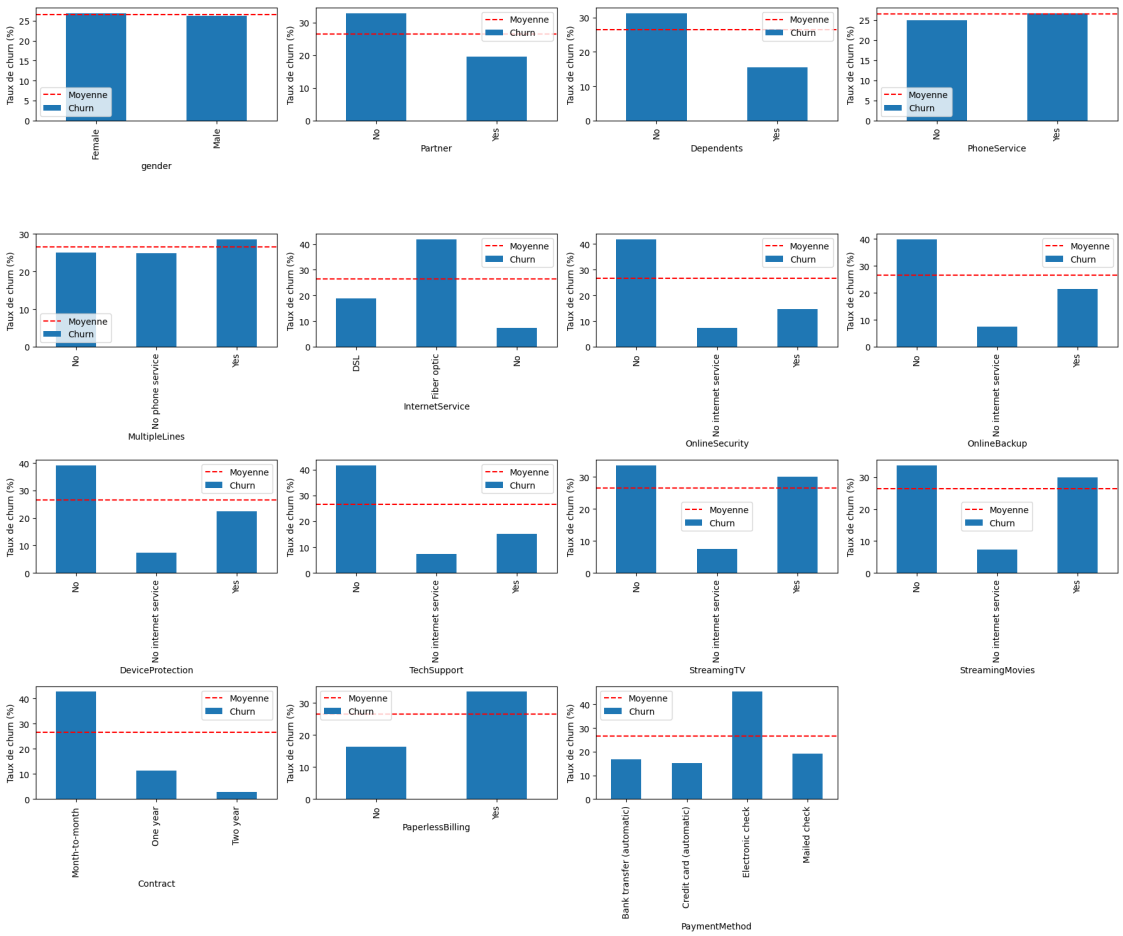
Les clients avec des factures élevées partent davantage : soit le prix est perçu comme trop élevé ou une meilleure offre tarifaire est disponible chez la concurrence. Et les clients qui paient moins sont plus satisfaits du rapport qualité/prix.

Graphique 3 : TotalCharges selon le churn

- Clients fidèles (No churn) : médiane ~ 1800
- Clients partis (Yes churn) : médiane ~ 1000$

Les clients qui partent n'ont pas eu le temps de générer beaucoup de revenus.

[1253]:
```
# Distribution variable catégorielles et churn
cols_cat = ['gender', 'Partner', 'Dependents', 'PhoneService','MultipleLines',␣
 ↪'InternetService', 'OnlineSecurity', 'OnlineBackup',
        'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',␣
 ↪'Contract', 'PaperlessBilling', 'PaymentMethod']
plt.figure(figsize=(18,15))
for i, col in enumerate (cols_cat, 1):
    churn_rate = data_clean.groupby(col)['Churn'].apply(lambda x: (x =='Yes').
 ↪sum()/len(x) * 100)
    plt.subplot(4,4,i)
    churn_rate.plot(kind='bar', x=col)
    plt.xlabel(col)
    plt.ylabel('Taux de churn (%)')
    plt.axhline(y=26.5, color='red', linestyle= '--', label='Moyenne')
    plt.legend()
plt.tight_layout()
plt.show()
```



Le type de contrat apparaît comme le facteur le plus discriminant avec un contraste saisissant entre les différentes formules. Les clients en contrat mensuel sans engagement présentent un taux de churn catastrophique de 43%, soit environ quatorze fois supérieur aux clients engagés sur deux ans qui n'affichent qu'un taux de 3%. Cette différence massive suggère que l'absence d'engagement contractuel facilite grandement le départ des clients insatisfaits, tandis que l'engagement long terme crée une barrière à la sortie et incite probablement les clients à donner une seconde chance au service.

La méthode de paiement constitue le second facteur critique avec des écarts tout aussi spectaculaires. Les clients payant par chèque électronique affichent un taux de churn de 45%, soit trois fois supérieur aux clients utilisant les prélèvements automatiques bancaires ou par carte de crédit qui tournent autour de 15-17%. Ce résultat suggère fortement que l'automatisation du paiement réduit considérablement le risque de départ, probablement en créant une friction supplémentaire pour l'annulation et en évitant les oublis de paiement qui peuvent déclencher des résiliations.

Le type de service internet révèle un problème majeur concernant l'offre fibre optique qui enregistre un taux de churn alarmant de 42%, soit plus du double du taux observé pour le DSL à 19%. Cette situation est particulièrement préoccupante car la fibre optique représente généralement l'offre premium de l'entreprise. Les raisons possibles incluent un prix perçu comme trop élevé par rapport à la valeur délivrée, des problèmes techniques de qualité de service, ou une concurrence particulièrement

agressive sur ce segment haut de gamme.

Les services additionnels jouent un rôle protecteur majeur contre le churn. Les clients ne disposant pas de support technique, de sécurité en ligne, de sauvegarde cloud ou de protection des appareils présentent des taux de churn systématiquement supérieurs à 39-42%, tandis que ceux bénéficiant de ces services descendent à environ 15%. Cette corrélation forte suggère que ces services créent de la valeur perçue pour le client et renforcent son attachement à l'opérateur, justifiant ainsi une stratégie de bundling pour augmenter la rétention.

La situation familiale du client influence également significativement le comportement de churn. Les clients célibataires sans conjoint affichent un taux de churn de 33% contre 20% pour ceux en couple, tandis que l'absence de personnes à charge fait grimper le taux à 31% contre seulement 15% pour les clients avec des enfants. Ces chiffres suggèrent que les clients avec des responsabilités familiales recherchent davantage la stabilité et sont moins enclins à changer de fournisseur, probablement pour éviter les désagréments d'un changement de service pour l'ensemble du foyer.

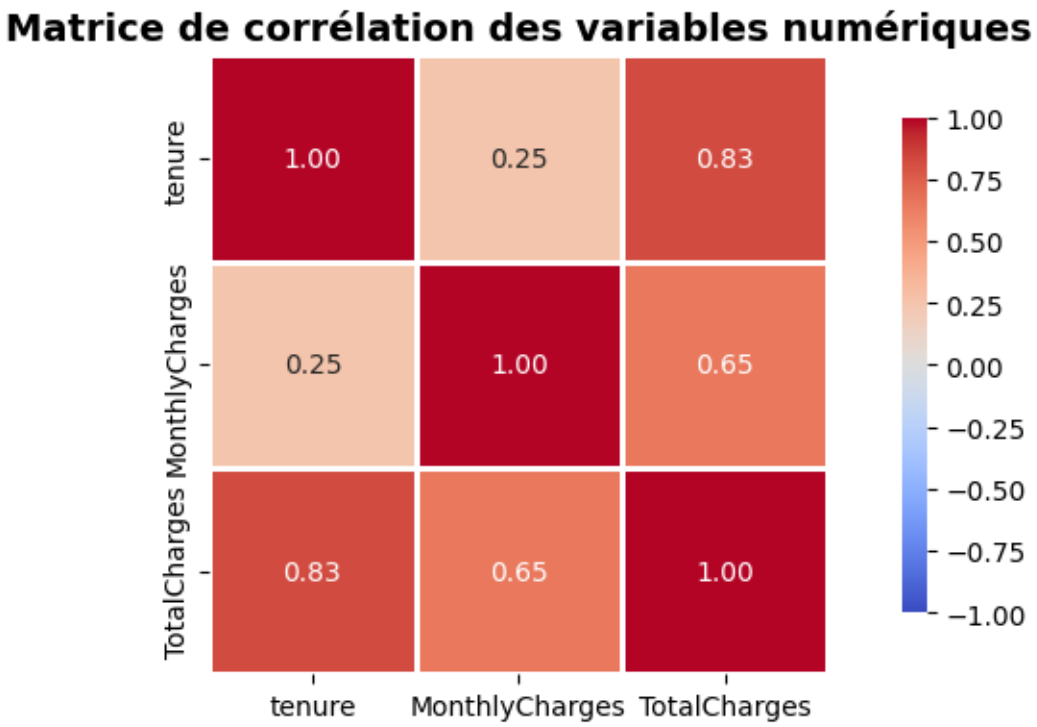
La facturation électronique présente une corrélation contre-intuitive avec un taux de churn de 33% pour les clients optant pour cette option moderne contre 16% pour ceux conservant la facture papier. Ce résultat peut s'expliquer par un effet de composition, les clients technophiles et digitaux étant généralement plus jeunes, plus volatiles et plus enclins à comparer les offres et changer de fournisseur facilement.

Enfin, certaines variables comme le genre et le service téléphonique ne montrent aucun impact significatif sur le churn, avec des taux pratiquement identiques entre les différentes catégories, autour de la moyenne générale de 26,5%. Ces variables pourront donc être considérées comme moins prioritaires dans la modélisation prédictive.

Le profil du client à très haut risque de churn combine plusieurs caractéristiques défavorables : un contrat mensuel sans engagement, un paiement par chèque électronique, un abonnement fibre optique sans services additionnels, un statut de célibataire sans enfants, une ancienneté récente et une facture mensuelle élevée.

6.4.3 *Analyse de corrélations*

[1256]:
```
cols_num = ['tenure', 'MonthlyCharges', 'TotalCharges']
correlation = data_clean[cols_num].corr()
plt.figure(figsize=(8, 4))
sns.heatmap(correlation, annot=True, fmt='.2f', cmap='coolwarm',
            center=0, square=True, linewidths=2, cbar_kws={"shrink": 0.8},
            vmin=-1, vmax=1)
plt.title('Matrice de corrélation des variables numériques', fontsize=14,␣
 ↪weight='bold')
plt.tight_layout()
plt.show()
```



L'analyse de la matrice de corrélation entre les trois variables numériques révèle des relations intéressantes. La corrélation la plus forte et la plus préoccupante concerne la relation entre l'ancienneté du client et le montant total facturé, avec un coefficient de corrélation de 0.83. Cette valeur élevée s'explique parfaitement par la logique métier puisque le montant total facturé correspond mathématiquement à l'accumulation des factures mensuelles sur la durée de l'abonnement, créant ainsi une relation quasi-linéaire entre ces deux variables. Cette forte multicolinéarité pose un problème potentiel pour la modélisation car ces deux variables apportent essentiellement la même information sous des formes différentes, ce qui pourrait créer de l'instabilité dans les coefficients du modèle et compliquer l'interprétation de l'importance relative de chaque variable.

La corrélation entre les charges mensuelles et le total facturé atteint 0.65, ce qui représente une corrélation modérée mais moins problématique que la précédente. Cette relation s'explique par le fait que les clients ayant des factures mensuelles élevées accumulent naturellement des montants totaux plus importants au fil du temps, mais cette relation n'est pas parfaitement linéaire car elle dépend également de l'ancienneté. Les clients récents avec des factures élevées n'auront pas encore accumulé un montant total important, ce qui explique pourquoi la corrélation reste modérée plutôt que forte.

En revanche, la corrélation entre l'ancienneté et les charges mensuelles est très faible avec un coefficient de seulement 0.25. Cette faible corrélation indique que ces deux variables sont largement indépendantes l'une de l'autre et apportent donc des informations complémentaires et non redondantes au modèle prédictif. Cela signifie qu'un client peut avoir une longue ancienneté avec une facture mensuelle modeste tout comme un nouveau client peut avoir une facture mensuelle élevée, et vice versa.

Lors de la phase de préparation des données pour la modélisation, je supprimerai la variable TotalCharges du jeu de données afin d'éviter les problèmes de multicolinéarité. La variable tenure sera conservée car elle apporte une information plus directement pertinente pour prédire le churn, à savoir l'ancienneté du client qui s'est révélée être un facteur discriminant majeur dans l'analyse bivariée.

6.5 *Etape 4 : Feature engineering*

[1259]:
```
data_eng = data_clean.copy()
```

6.5.1 *Catégories d'ancienneté*

[1261]:
```
def cat_tenure(tenure):
    if tenure <= 12:
        return 'New'
    elif tenure <= 24:
        return 'Medium'
    elif tenure <= 49:
        return 'Long'
    else:
        return 'Very long'
data_eng['tenure_group'] = data_eng['tenure'].apply(cat_tenure)
```

[1262]:
```
# Distribution
data_eng['tenure_group'].value_counts()
```

[1262]:
```
tenure_group
New          2186
Very long    2173
Long         1660
Medium       1024
Name: count, dtype: int64
```

[1263]:
```
# Taux de churn par groupe
churn_rate_TG = data_eng.groupby('tenure_group')['Churn'].apply(lambda x : (x␣
 ↪=='Yes').sum() / len(x) * 100)
churn_rate_TG
```

[1263]:
```
tenure_group
Long         20.481928
Medium       28.710938
New          47.438243
Very long     9.111827
Name: Churn, dtype: float64
```

6.5.2 *Nombre de services souscrits*

[1265]:
```
# Liste des services additionnels
services = ['OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
            'TechSupport', 'StreamingTV', 'StreamingMovies']
```

[1266]:
```
# Nouvelle variable
data_eng['Totalservice'] = (data_eng[services] == 'Yes').sum(axis=1)
```

[1267]:
```
# Distribution
data_eng['Totalservice'].value_counts()
```

[1267]:
```
Totalservice
0    2219
3    1118
2    1033
1     966
4     852
5     571
6     284
Name: count, dtype: int64
```

[1268]:
```
# Taux de churn
churn_rate_TS = data_eng.groupby('Totalservice')['Churn'].apply(lambda x: (x ==␣
 ↪'Yes').sum() / len(x) * 100)
churn_rate_TS
```

[1268]:
```
Totalservice
0    21.406039
1    45.755694
2    35.818006
3    27.370304
4    22.300469
5    12.434326
6     5.281690
Name: Churn, dtype: float64
```

6.5.3 *Clients à haut risque*

[1270]:
```
# Flag pour les clients à très haut risque
data_eng['Risk_profile'] = (
    (data_eng['Contract'] == 'Month-to-month') &
    (data_eng['tenure'] <= 12) &
    (data_eng['PaymentMethod'] == 'Electronic check')
).astype(int)
```

[1271]:
```
# Distribution
data_eng['Risk_profile'].value_counts()
```

[1271]:
```
Risk_profile
0    6089
1     954
Name: count, dtype: int64
```

[1272]:
```
# Taux de churn
churn_rate_RP = data_eng.groupby('Risk_profile')['Churn'].apply(lambda x: (x ==␣
 ↪'Yes').sum() / len(x) * 100)
churn_rate_RP
```

[1272]:
```
Risk_profile
0    20.808014
1    63.102725
Name: Churn, dtype: float64
```

6.5.4 *Paiement automatique*

[1274]:
```
# Indicateur
data_eng['Auto_payment'] = data_eng['PaymentMethod'].isin([
    'Bank transfer (automatic)',
    'Credit card (automatic)'
]).astype(int)
```

[1275]:
```
# Distribution
data_eng['Auto_payment'].value_counts()
```

[1275]:
```
Auto_payment
0    3977
1    3066
Name: count, dtype: int64
```

[1276]:
```
# Taux de churn
churn_rate_AP = data_eng.groupby('Auto_payment')['Churn'].apply(lambda x: (x ==␣
 ↪'Yes').sum() / len(x) * 100)
churn_rate_AP
```

[1276]:
```
Auto_payment
0    34.674378
1    15.981735
Name: Churn, dtype: float64
```

6.5.5 *Ratio MonthlyCharges par service*

[1278]:
```
# Ratio charges mensuelles / nombre de services
data_eng['Charges_par_service'] = data_eng['MonthlyCharges'] /␣
 ↪(data_eng['Totalservice'] + 1)
```

[1279]:
```
# Statistiques
data_eng['Charges_par_service'].describe()
```

[1279]:
```
count    7043.000000
mean       25.490020
std        14.093315
min         8.650000
25%        18.025000
50%        20.660000
75%        27.425000
max        77.900000
Name: Charges_par_service, dtype: float64
```

[1280]:
```
# Relation avec le churn
data_eng.groupby('Churn')['Charges_par_service'].describe()
```

[1280]:

Churn	count	mean	std	min	25%	50%	75%	max
No	5174.0	22.785474	11.498748	8.74	16.60	20.05	24.85	77.90
Yes	1869.0	32.977082	17.500354	8.65	20.68	26.70	40.20	77.15

L'analyse des résultats de la variable Charges_par_service révèle une validation claire de l'hypothèse initiale, avec les clients qui partent affichant un ratio moyen de 32.98 dollars par service contre seulement 22.79 pour les clients fidèles, confirmant que ceux qui perçoivent un mauvais rapport qualité-prix ont effectivement tendance à churner davantage.

La création des catégories d'ancienneté via tenure_group révèle un gradient de churn spectaculaire allant de 47% pour les nouveaux clients à seulement 9% pour les très anciens. Le comptage des services additionnels via TotalServices démontre une relation inverse claire avec le churn, passant de 46% pour les clients avec un seul service à seulement 5% pour ceux bénéficiant des six services, avec la découverte intéressante qu'avoir un unique service est paradoxalement pire que n'en avoir aucun, suggérant un possible effet de déception ou d'insatisfaction partielle.

Le flag de profil à haut risque Risk_profile révèle un taux de churn catastrophique de 63% pour les clients combinant contrat mensuel, faible ancienneté et paiement par chèque électronique, contre seulement 21% pour les autres profils, offrant ainsi au modèle une variable synthétique extrêmement prédictive. L'indicateur de paiement automatique Auto_payment confirme l'impact majeur du mode de paiement avec une réduction de plus de 50% du taux de churn entre les clients sans automatisation (35%) et ceux avec prélèvement automatique (16%), représentant un levier d'action business concret et facilement déployable.

6.6 *Etape 5 : Préparation des données pour la modélisation*

6.6.1 *Nettoyage*

[1284]:
```
data_model = data_eng.copy()
```

[1285]:
```
# Variables à exclure
columns_to_drop = ['customerID', 'TotalCharges']
```

```
data_model = data_model.drop(columns=columns_to_drop)
```

[1286]:
```
# Dimensions
data_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   gender               7043 non-null   object
 1   SeniorCitizen        7043 non-null   int64
 2   Partner              7043 non-null   object
 3   Dependents           7043 non-null   object
 4   tenure               7043 non-null   int64
 5   PhoneService         7043 non-null   object
 6   MultipleLines        7043 non-null   object
 7   InternetService      7043 non-null   object
 8   OnlineSecurity       7043 non-null   object
 9   OnlineBackup         7043 non-null   object
 10  DeviceProtection     7043 non-null   object
 11  TechSupport          7043 non-null   object
 12  StreamingTV          7043 non-null   object
 13  StreamingMovies      7043 non-null   object
 14  Contract             7043 non-null   object
 15  PaperlessBilling     7043 non-null   object
 16  PaymentMethod        7043 non-null   object
 17  MonthlyCharges       7043 non-null   float64
 18  Churn                7043 non-null   object
 19  tenure_group         7043 non-null   object
 20  Totalservice         7043 non-null   int64
 21  Risk_profile         7043 non-null   int64
 22  Auto_payment         7043 non-null   int64
 23  Charges_par_service  7043 non-null   float64
dtypes: float64(2), int64(5), object(17)
memory usage: 1.3+ MB
```

6.6.2 *Encodage*

Churn

[1289]:
```
data_model['Churn'] = data_model['Churn'].map({'Yes' : 1, 'No' : 0})
```

[1290]:
```
data_model['Churn'].value_counts()
```

[1290]:
```
Churn
0    5174
1    1869
```

```
Name: count, dtype: int64
```

Autres variables catégorielles

[1292]:
```
# Liste des variables catégorielles
vars_cat = data_model.select_dtypes(include='object').columns.tolist()
vars_cat
```

[1292]:
```
['gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'tenure_group']
```

[1293]:
```
# Encodage
data_model = pd.get_dummies(data_model, columns = vars_cat, drop_first=True)
```

[1294]:
```
data_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 37 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   SeniorCitizen                          7043 non-null   int64
 1   tenure                                 7043 non-null   int64
 2   MonthlyCharges                         7043 non-null   float64
 3   Churn                                  7043 non-null   int64
 4   Totalservice                           7043 non-null   int64
 5   Risk_profile                           7043 non-null   int64
 6   Auto_payment                           7043 non-null   int64
 7   Charges_par_service                    7043 non-null   float64
 8   gender_Male                            7043 non-null   bool
 9   Partner_Yes                            7043 non-null   bool
 10  Dependents_Yes                         7043 non-null   bool
 11  PhoneService_Yes                       7043 non-null   bool
 12  MultipleLines_No phone service         7043 non-null   bool
 13  MultipleLines_Yes                      7043 non-null   bool
```

```
 14  InternetService_Fiber optic            7043 non-null   bool
 15  InternetService_No                     7043 non-null   bool
 16  OnlineSecurity_No internet service     7043 non-null   bool
 17  OnlineSecurity_Yes                     7043 non-null   bool
 18  OnlineBackup_No internet service       7043 non-null   bool
 19  OnlineBackup_Yes                       7043 non-null   bool
 20  DeviceProtection_No internet service   7043 non-null   bool
 21  DeviceProtection_Yes                   7043 non-null   bool
 22  TechSupport_No internet service        7043 non-null   bool
 23  TechSupport_Yes                        7043 non-null   bool
 24  StreamingTV_No internet service        7043 non-null   bool
 25  StreamingTV_Yes                        7043 non-null   bool
 26  StreamingMovies_No internet service    7043 non-null   bool
 27  StreamingMovies_Yes                    7043 non-null   bool
 28  Contract_One year                      7043 non-null   bool
 29  Contract_Two year                      7043 non-null   bool
 30  PaperlessBilling_Yes                   7043 non-null   bool
 31  PaymentMethod_Credit card (automatic)  7043 non-null   bool
 32  PaymentMethod_Electronic check         7043 non-null   bool
 33  PaymentMethod_Mailed check             7043 non-null   bool
 34  tenure_group_Medium                    7043 non-null   bool
 35  tenure_group_New                       7043 non-null   bool
 36  tenure_group_Very long                 7043 non-null   bool
dtypes: bool(29), float64(2), int64(6)
memory usage: 639.8 KB
```

6.6.3 *Données de test et d'entrainement*

[1296]:
```
# Variables (expliquée et explicatives)
X = data_model.drop('Churn', axis=1)
y = data_model['Churn']
```

[1297]:
```
# Données de test et d'entraînement
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

6.6.4 *Scaling*

[1299]:
```
scaler = StandardScaler()
```

[1300]:
```
# Variables numériques
vars_num = ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'Totalservice',␣
 ↪'Risk_profile', 'Auto_payment', 'Charges_par_service']
```

[1301]:
```
# Fit sur train
scaler.fit(X_train[vars_num])
```

[1301]:
```
StandardScaler()
```

[1302]:
```
# Transformation
X_train[vars_num] = scaler.transform(X_train[vars_num])
X_test[vars_num] = scaler.transform(X_test[vars_num])
```

[1303]:
```
X_train[vars_num].head()
```

[1303]:
```
      SeniorCitizen    tenure  MonthlyCharges  Totalservice  Risk_profile  \
3738      -0.441773  0.102371       -0.521976      0.507935     -0.395779
3151      -0.441773 -0.711743        0.337478     -0.570530     -0.395779
4860      -0.441773 -0.793155       -0.809013      0.507935     -0.395779
3867      -0.441773 -0.263980        0.284384      1.047168     -0.395779
3810      -0.441773 -1.281624       -0.676279     -1.109762      2.526660

      Auto_payment  Charges_par_service
3738     -0.879415            -0.941984
3151     -0.879415             0.881851
4860     -0.879415            -1.098183
3867      1.137119            -0.768629
3810     -0.879415             1.387469
```

6.7 *Etape 6 : Modélisation*

6.7.1 *Regression logistique*

[1306]:
```
# Entraînement du modèle
log_reg = LogisticRegression(random_state=42, max_iter=1000)
```

[1307]:
```
log_reg.fit(X_train, y_train)
```

[1307]:
```
LogisticRegression(max_iter=1000, random_state=42)
```

[1308]:
```
# Prédictions
y_pred_lr = log_reg.predict(X_test)
y_pred_proba_lr = log_reg.predict_proba(X_test) [:,1]
```

[1309]:
```
# Évaluation
print(f"Accuracy : {accuracy_score(y_test, y_pred_lr):.4f}")
print(f"ROC-AUC Score : {roc_auc_score(y_test, y_pred_proba_lr):.4f}")
print("\nRapport de classification :")
print(classification_report(y_test, y_pred_lr, target_names=['No Churn',␣
 ↪'Churn']))
```

```
Accuracy : 0.7956
ROC-AUC Score : 0.8418
```

```
Rapport de classification :
              precision    recall  f1-score   support

    No Churn       0.84      0.90      0.87      1035
       Churn       0.64      0.52      0.57       374

    accuracy                           0.80      1409
   macro avg       0.74      0.71      0.72      1409
weighted avg       0.79      0.80      0.79      1409
```

[1310]:
```
# Matrice de confusion
print("Matrice de confusion")
cm_lr = confusion_matrix(y_test, y_pred_lr)
cm_lr
```

```
Matrice de confusion
```

[1310]:
```
array([[927, 108],
       [180, 194]])
```

Le modèle de régression logistique affiche des performances globalement honorables avec une accuracy de 79,6% et un score ROC-AUC de 84,2%, démontrant une bonne capacité à discriminer entre les clients fidèles et ceux à risque de churn. Le modèle excelle particulièrement dans l'identification des clients fidèles avec un recall de 90%, ce qui signifie qu'il détecte correctement neuf clients sur dix qui resteront dans l'entreprise. Cependant, le problème de ce modèle réside dans sa performance sur la classe churn avec un recall critique de seulement 52%, ce qui se traduit concrètement par 180 clients sur 374 qui vont effectivement partir mais que le modèle ne parvient pas à identifier. Je vais explorer d'autres modèles capables d'améliorer ce recall.

6.7.2 *Random forest*

[1313]:
```
# Modèle
rf_model = RandomForestClassifier(
    n_estimators=100,
    random_state=42,
    class_weight='balanced',
    max_depth=10,
    n_jobs=-1
)
```

[1314]:
```
# Fit
rf_model.fit(X_train, y_train)
```

[1314]:
```
RandomForestClassifier(class_weight='balanced', max_depth=10, n_jobs=-1,
                       random_state=42)
```

[1315]:
```
# Prédictions
y_pred_rf = rf_model.predict(X_test)
y_pred_proba_rf = rf_model.predict_proba(X_test)[:,1]
```

[1316]:
```
# Évaluation
print(f"Accuracy : {accuracy_score(y_test, y_pred_rf):.4f}")
print(f"ROC-AUC Score : {roc_auc_score(y_test, y_pred_proba_rf):.4f}")
print("\nRapport de classification :")
print(classification_report(y_test, y_pred_rf, target_names=['No Churn',␣
 ↪'Churn']))
```

```
Accuracy : 0.7622
ROC-AUC Score : 0.8387

Rapport de classification :
              precision    recall  f1-score   support

    No Churn       0.88      0.78      0.83      1035
       Churn       0.54      0.72      0.62       374

    accuracy                           0.76      1409
   macro avg       0.71      0.75      0.72      1409
weighted avg       0.79      0.76      0.77      1409
```

[1317]:
```
# Matrice de confusion
print("\nMatrice de confusion :")
cm_rf = confusion_matrix(y_test, y_pred_rf)
cm_rf
```

```
Matrice de confusion :
```

[1317]:
```
array([[806, 229],
       [106, 268]])
```

Le modèle Random Forest démontre une amélioration dans la détection des clients à risque de churn en atteignant un recall de 72% sur la classe minoritaire, soit une progression spectaculaire de 20 points par rapport à la régression logistique qui plafonnait à 52%. Cette performance se traduit concrètement par l'identification correcte de 268 clients churn sur 374, ne laissant que 106 clients à risque non détectés contre 180 précédemment, ce qui représente une réduction de 41% des faux négatifs et donc 74 clients supplémentaires qui pourront bénéficier d'actions de rétention ciblées.

Le score ROC-AUC reste excellent à 83,9% mais on note une légère baisse de 3,4 points de l'accuracy par rapport à LR.

6.7.3 *XGBoost*

[1320]:
```
# Ratio de déséquilibre
scale_pos_weight = (y_train == 0).sum() / (y_train == 1).sum()
print(f"Ratio de déséquilibre : {scale_pos_weight:.2f}")
```

```
Ratio de déséquilibre : 2.77
```

[1321]:
```
# Entraînement
xgb_model = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    scale_pos_weight=scale_pos_weight,  # Gérer le déséquilibre
    random_state=42,
    eval_metric='logloss'
)
```

[1322]:
```
# Fit
xgb_model.fit(X_train, y_train)
```

[1322]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.1, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=100,
              n_jobs=None, num_parallel_tree=None, random_state=42, …)
```

[1323]:
```
# Prédictions
y_pred_xgb = xgb_model.predict(X_test)
y_pred_proba_xgb = xgb_model.predict_proba(X_test)[:, 1]
```

[1324]:
```
# Évaluation
print(f"Accuracy : {accuracy_score(y_test, y_pred_xgb):.4f}")
print(f"ROC-AUC Score : {roc_auc_score(y_test, y_pred_proba_xgb):.4f}")
print("\nRapport de classification :")
print(classification_report(y_test, y_pred_xgb, target_names=['No Churn',␣
↪'Churn']))
```

```
Accuracy : 0.7559
ROC-AUC Score : 0.8378

Rapport de classification :
              precision    recall  f1-score   support
```

```
    No Churn       0.90      0.75      0.82      1035
       Churn       0.53      0.77      0.63       374

    accuracy                           0.76      1409
   macro avg       0.71      0.76      0.72      1409
weighted avg       0.80      0.76      0.77      1409
```

[1325]:
```
# Matrice de confusion
print("\nMatrice de confusion :")
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
cm_xgb
```

```
Matrice de confusion :
```

[1325]:
```
array([[778, 257],
       [ 87, 287]])
```

Le modèle XGBoost s'impose comme le champion de la détection du churn avec un recall de 77% sur la classe minoritaire, surpassant Random Forest de 5 points et la régression logistique de 25 points, ce qui se traduit par l'identification correcte de 287 clients à risque sur 374, ne laissant que 87 clients churn non détectés contre 106 pour Random Forest et 180 pour la régression logistique. Cette performance exceptionnelle représente une amélioration business majeure avec 93 clients supplémentaires sauvés par rapport au modèle baseline et 19 clients de plus par rapport à Random Forest, justifiant pleinement l'utilisation du paramètre scale_pos_weight pour gérer efficacement le déséquilibre de classes.

6.7.4 *Récapitulatif évaluation modèles*

Métrique	Logistic Regression	Random Forest	XGBoost
Accuracy	79.6%	76.2%	75.6%
ROC-AUC	84.2%	83.9%	83.8%
Recall Churn	52%	72%	77%
Precision Churn	64%	54%	53%
F1-Score Churn	57%	62%	63%
Clients churn détectés	194/374	268/374	287/374
Clients churn manqués (FN)	180	106	87
Fausses alertes (FP)	108	229	257

6.8 *Etape 7 : Optimisation*

[1330]:
```
# Calcul scale_pos_weight
scale_pos_weight = (y_train == 0).sum() / (y_train == 1).sum()
```

[1331]:
```
# Paramètres
param_grid = {
    'max_depth': [4, 6, 8],
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.8, 1.0]
}
```

[1332]:
```
# Modèle
xgb_base = XGBClassifier(
    scale_pos_weight=scale_pos_weight,
    random_state=42,
    eval_metric='logloss'
)
```

[1333]:
```
# GridSearchCV
grid_search = GridSearchCV(
    estimator=xgb_base,
    param_grid=param_grid,
    scoring='recall',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

[1334]:
```
# Entraînement
grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
```

[1334]:
```
GridSearchCV(cv=3,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, device=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False,
                                     eval_metric='logloss', feature_types=None,
                                     gamma=None, grow_policy=None,
                                     importance_type=None,
                                     interaction_constraints=None,
                                     learning_rate=…
                                     max_delta_step=None, max_depth=None,
                                     max_leaves=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     multi_strategy=None, n_estimators=None,
                                     n_jobs=None, num_parallel_tree=None,
```

```
                                random_state=42, …),
            n_jobs=-1,
            param_grid={'learning_rate': [0.01, 0.1, 0.2],
                        'max_depth': [4, 6, 8], 'min_child_weight': [1, 3, 5],
                        'n_estimators': [100, 200], 'subsample': [0.8, 1.0]},
            scoring='recall', verbose=1)
```

[1335]:
```
print(f"\nMeilleurs paramètres trouvés :")
print(grid_search.best_params_)
print(f"\nMeilleur score CV (Recall) : {grid_search.best_score_:.4f}")
```

```
Meilleurs paramètres trouvés :
{'learning_rate': 0.01, 'max_depth': 4, 'min_child_weight': 1, 'n_estimators':
100, 'subsample': 1.0}

Meilleur score CV (Recall) : 0.8181
```

[1336]:
```
# Meilleur modèle
best_xgb = grid_search.best_estimator_
```

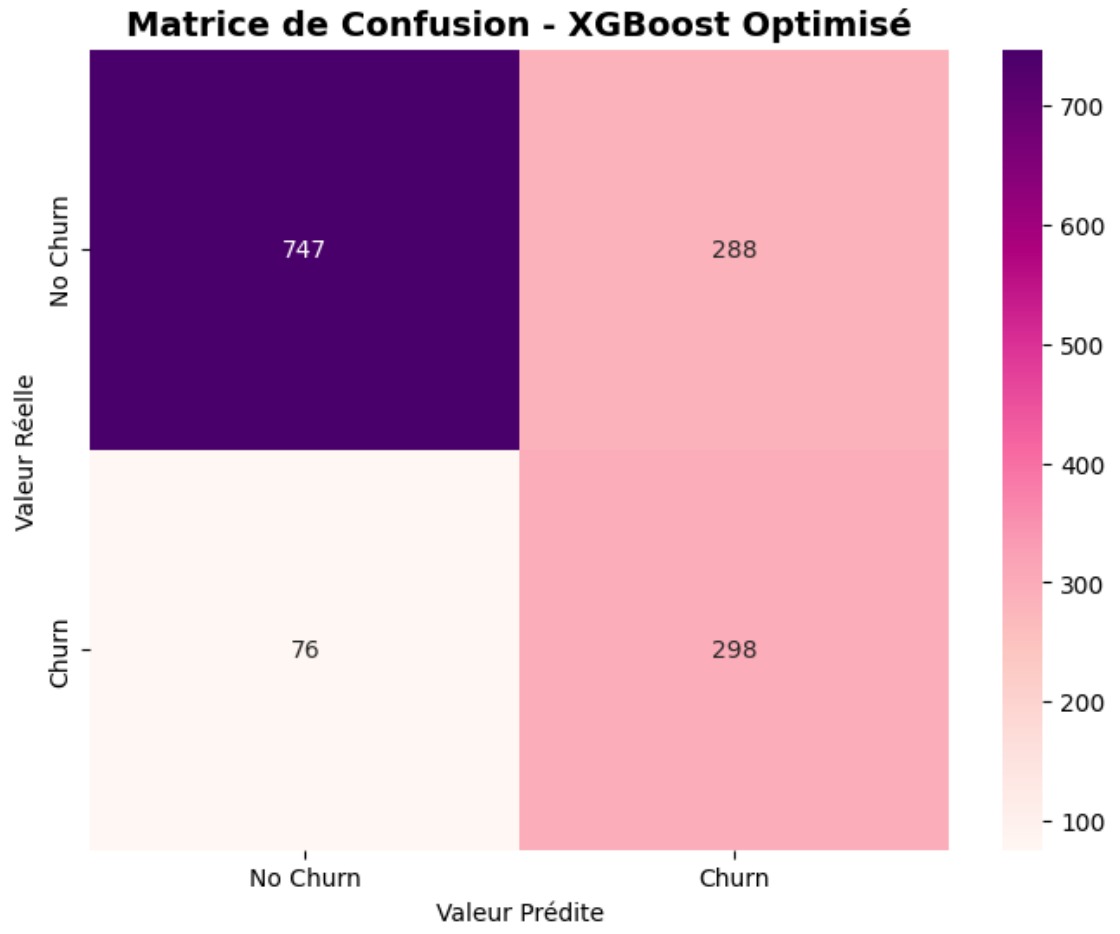
[1337]:
```
# Prédictions avec le modèle optimisé
y_pred_best = best_xgb.predict(X_test)
y_pred_proba_best = best_xgb.predict_proba(X_test)[:, 1]
```

[1338]:
```
# Évaluation
print(f"\nAccuracy : {accuracy_score(y_test, y_pred_best):.4f}")
print(f"ROC-AUC Score : {roc_auc_score(y_test, y_pred_proba_best):.4f}")
print("\nRapport de classification :")
print(classification_report(y_test, y_pred_best, target_names=['No Churn',␣
 ↪'Churn']))
```

```
Accuracy : 0.7417
ROC-AUC Score : 0.8358

Rapport de classification :
              precision    recall  f1-score   support

    No Churn       0.91      0.72      0.80      1035
       Churn       0.51      0.80      0.62       374

    accuracy                           0.74      1409
   macro avg       0.71      0.76      0.71      1409
weighted avg       0.80      0.74      0.76      1409
```

[1339]:
```
# Matrice de confusion
cm_best = confusion_matrix(y_test, y_pred_best)
tn, fp, fn, tp = cm_best.ravel()
```

[1340]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm_best, annot=True, fmt='d', cmap='RdPu',
            xticklabels=['No Churn', 'Churn'],
            yticklabels=['No Churn', 'Churn'])
plt.title('Matrice de Confusion - XGBoost Optimisé', fontsize=14, weight='bold')
plt.ylabel('Valeur Réelle')
plt.xlabel('Valeur Prédite')
plt.show()
```



Métrique	Logistic Regression	Random Forest	XGBoost	XGBoost Opti
Accuracy	79.6%	76.2%	75.6%	74.2%
ROC-AUC	84.2%	83.9%	83.8%	83.6%

```
Recall Churn                |        52%          |      72%        |     77%       |         80%
Precision Churn             |        64%          |      54%        |     53%       |         51%
F1-Score Churn              |        57%          |      62%        |     63%       |         62%
Clients churn détectés      |      194/374        |    268/374      |   287/374     |       298/374
Clients churn manqués (FN)  |        180          |      106        |     87        |         76
Fausses alertes (FP)        |        108          |      229        |     257       |         288
```

6.9 *Etape 8 : Interprétation du modèle*

[1343]:
```
# Récupérer l'importance des features
feature_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': best_xgb.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nTop 15 variables les plus importantes :")
print(feature_importance.head(15))
```

```
Top 15 variables les plus importantes :
                          Feature  Importance
28               Contract_Two year    0.374802
27               Contract_One year    0.239158
13    InternetService_Fiber optic    0.119556
4                     Risk_profile    0.046619
14              InternetService_No    0.046090
26            StreamingMovies_Yes    0.036900
1                          tenure    0.030638
24                StreamingTV_Yes    0.029887
10                PhoneService_Yes    0.026372
31  PaymentMethod_Electronic check    0.013543
2                   MonthlyCharges    0.008754
6               Charges_par_service    0.007875
29            PaperlessBilling_Yes    0.007377
5                     Auto_payment    0.007198
7                      gender_Male    0.004513
```

L'analyse de l'importance des variables révèle une hiérarchie très claire des facteurs de churn, dominée par le type de contrat qui représente à lui seul 61,4% de l'importance totale du modèle avec Contract_Two year comptant pour 37,5% et Contract_One year pour 23,9%, confirmant ainsi que l'engagement contractuel constitue le levier de rétention le plus puissant à la disposition de l'entreprise.

Le type de service internet arrive en deuxième position avec InternetService_Fiber optic pesant 12% de l'importance, validant notre découverte lors de l'analyse exploratoire que l'offre fibre optique souffre d'un problème majeur de qualité ou de perception de valeur qui pousse 42% de ses abonnés au départ.

Le feature engineering trouve une validation éclatante avec Risk_profile qui se positionne au qua-

trième rang avec 4,7% d'importance, démontrant que cette variable synthétique capturant simultanément le contrat mensuel, la faible ancienneté et le paiement par chèque électronique apporte une valeur prédictive significative au modèle et confirme l'existence d'un profil client à très haut risque clairement identifiable.

Les services additionnels comme StreamingMovies et StreamingTV apparaissent en bonne position avec respectivement 3,7% et 3% d'importance, tandis que les variables traditionnelles comme tenure (3,1%) et MonthlyCharges (0,9%) jouent un rôle plus modeste que ce que leur importance dans l'analyse bivariée pouvait laisser présager, suggérant que leur effet est largement capturé par les variables dérivées et les interactions entre features.

Cette hiérarchie d'importance dessine clairement les priorités d'action business avec en tête la migration des clients vers des contrats d'engagement, suivie de la résolution urgente des problèmes liés à l'offre fibre optique, puis l'enrichissement de l'offre par des bundles de services additionnels attractifs, et enfin un focus particulier sur l'onboarding et l'accompagnement des nouveaux clients durant leurs premiers mois critiques.

6.10 *Etape 9 : Validation du modèle*

[1346]:
```
# Variables explicatives et expliquée
X_all = data_model.drop('Churn', axis=1)
y_all = data_model['Churn']
```

[1347]:
```
X_all_scaled = X_all.copy()
numeric_features = ['SeniorCitizen', 'tenure', 'MonthlyCharges', 'Totalservice',
                    'Risk_profile', 'Auto_payment', 'Charges_par_service']

X_all_scaled[numeric_features] = scaler.transform(X_all[numeric_features])
```

[1348]:
```
# Prédiction
churn_probabilities = best_xgb.predict_proba(X_all_scaled)[:, 1] * 100
churn_predictions = best_xgb.predict(X_all_scaled)
```

[1349]:
```
# Création d'un dataset opérationnel
key_features = ['Contract', 'tenure', 'InternetService', 'PaymentMethod',
                'MonthlyCharges', 'Totalservice', 'Partner', 'Dependents']

client_data = data_eng[key_features].copy()
```

[1350]:
```
# Ajout des prédictions
client_data['Churn_Probability'] = churn_probabilities
client_data['Predicted_Churn'] = churn_predictions
client_data['Actual_Churn'] = y_all.map({1: 'Yes', 0: 'No'})
```

[1351]:
```
# Création d'un score de risque
def risk_category(prob):
    if prob >= 70:
        return 'CRITIQUE'
```

```
    elif prob >= 50:
        return 'ÉLEVÉ'
    elif prob >= 30:
        return 'MODÉRÉ'
    else:
        return 'FAIBLE'

client_data['Risk_Level'] = client_data['Churn_Probability'].
 ↪apply(risk_category)
```

[1352]:
```
client_data.head(5)
```

[1352]:

	Contract	tenure	InternetService	PaymentMethod	\
0	Month-to-month	1	DSL	Electronic check	
1	One year	34	DSL	Mailed check	
2	Month-to-month	2	DSL	Mailed check	
3	One year	45	DSL	Bank transfer (automatic)	
4	Month-to-month	2	Fiber optic	Electronic check	

	MonthlyCharges	Totalservice	Partner	Dependents	Churn_Probability	\
0	29.85	1	Yes	No	63.339161	
1	56.95	2	No	No	24.388273	
2	53.85	2	No	No	56.767536	
3	42.30	3	No	No	25.148014	
4	70.70	0	No	No	72.569008	

	Predicted_Churn	Actual_Churn	Risk_Level
0	1	No	ÉLEVÉ
1	0	No	FAIBLE
2	1	Yes	ÉLEVÉ
3	0	No	FAIBLE
4	1	Yes	CRITIQUE