NEOMA BUSINESS SCHOOL
REIMS · ROUEN · PARIS

Rapport du Churn sur le Dataset Telecom

2023

RIGAUD Eddy

Introduction

Objectifs :
"Prédire le Churn et analyser le comportement des clients pour les fidéliser."

Contenu :

Le jeu de données de churn Telecom, qui se compose de données d'activitées clients, ainsi qu'une étiquette de Churn spécifiant si un client a annulé son abonnement, sera utilisé pour développer des modèles prédictifs. L'ensemble de données a été télécharger sur Courses.

L'objectif de cette étude est d'analyser les données d'une entreprise de téléphonie mobile aux États-Unis afin de mieux comprendre le phénomène de churn et de proposer des actions à mettre en place pour réduire son impact sur l'entreprise.

Nous commencerons par une exploration des données pour avoir une vue d'ensemble sur les variables. Nous utiliserons également des techniques de visualisation pour mieux comprendre les données.
Ensuite, nous procéderons à une modélisation en utilisant des algorithmes d'apprentissage automatique pour prédire le churn et identifier les variables les plus importantes dans cette prédiction. Enfin, nous ferons des recommandations pour réduire le taux de churn en utilisant les résultats de notre analyse et de notre modélisation.

De plus, pour accompagner cette analyse, le dossier comprend également un Notebook d'analyse des données en Python ainsi que le code source de la page développée et déployée pour le rapport en question, accessible à l'adresse suivante :

https://hakini-telecom-churn-telecom-churn-app-tsudjz.streamlit.app/

Lien GitHub : https://github.com/HAKINI/Telecom_Churn
(De plus, tous les résultats présentés ici sont dans le Notebook présent dans le dossier du liens github avec le code python associé)

Page Streamlit

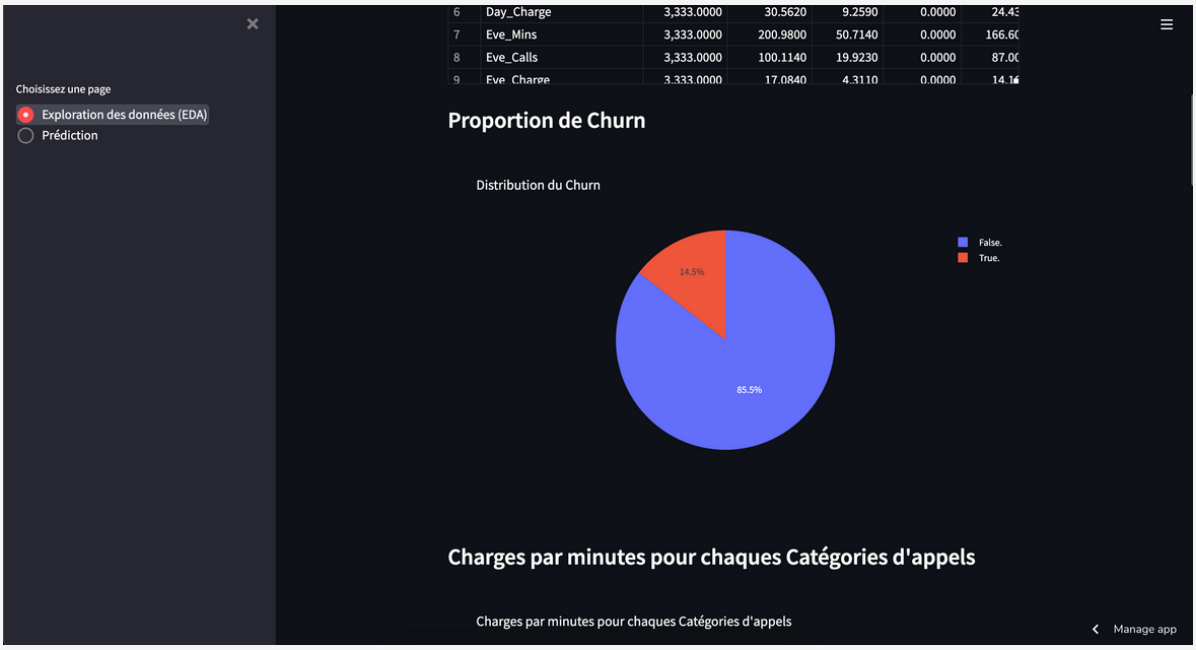
La page Streamlit est une interface utilisateur interactive qui permet d'analyser et de prédire le désabonnement des clients des télécommunications. Elle se compose de deux pages principales : Exploration des données (EDA) et Prédiction.

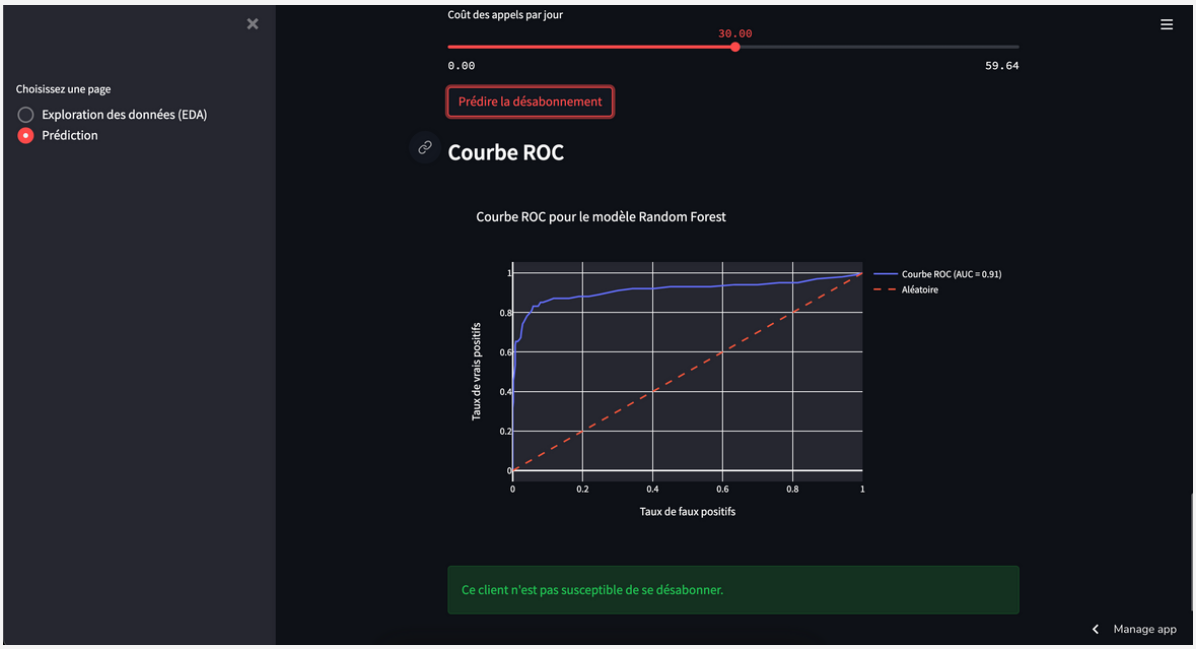
La page EDA présente un aperçu des données, des statistiques descriptives, des visualisations de la distribution du churn et des charges par minute pour chaque catégorie d'appels. Elle permet également d'explorer les histogrammes de Churn en fonction des Etats, du nombre d'appels au service client, de l'option Messagerie Vocale et de l'option appels à l'étranger. Enfin, elle présente une matrice de corrélation entre les variables et une répartition géographique des utilisateurs et du churn.

La page Prédiction permet de sélectionner un modèle de prédiction parmi Random Forest, Decision Tree, SVM et XGBoost. Elle présente ensuite un rapport de classification, une matrice de confusion et une courbe ROC pour évaluer la performance du modèle. Enfin, elle permet de saisir les informations d'un client spécifique et de prédire s'il est susceptible de se désabonner ou non en utilisant le modèle sélectionné.

Page Streamlit

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom





Data Cleaning

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom

Dans un premier temps, nous avons importé les libraires nécessaires pour notre analyse python, nous avons également regardé brièvement les données dans leurs ensemble. (Tout les screenshot, on était pris sur le Notebook Telecom_Churn_NB.ipynb dans le dossier remis)

Import Library

```
import plotly.express as px
import numpy as np # linear algebra
import pandas as pd # data processing
from math import * # module math
import matplotlib.pyplot as plt # visualization
# Visualisation
import plotly.offline as py # visualization
py.init_notebook_mode(connected=True) # visualization
import plotly.graph_objs as go # visualization
from plotly.subplots import make_subplots
import plotly.figure_factory as ff # visualization
import warnings
warnings.filterwarnings("ignore")

# Mapping
from geopy.geocoders import Nominatim
import folium
from folium.plugins import MarkerCluster
import time
from pycountry_convert import country_alpha2_to_continent_code, country_name_to_country_alpha2
from functools import partial

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
import xgboost as xgb
```

✓ 3.2s

On imprime les première ligne du Dataframe pour avoir une vision globale des données a traitées

```
df.head(5)
```

✓ 0.0s

	State	Account Length	Area Code	Phone	Int'l Plan	VMail Plan	VMail Message	Day Mins	Day Calls	Day Charge	...	Eve Calls	Eve Charge	Night Mins
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9

5 rows × 21 columns

Ensuite, vient la partie de nettoyages et la préparation des données.
Ici, on renomme les colonnes pour plus de clarté et pour faciliter le code (on insère des "_" pour lier 2 mots et éviter les espaces), on va calculer la proportion de répondants pour chaque État, la proportion de churn pour chaque État et on convertit les colonnes en type catégories. Pour cela, on combine toutes ces étapes dans la fonction prepare_data().

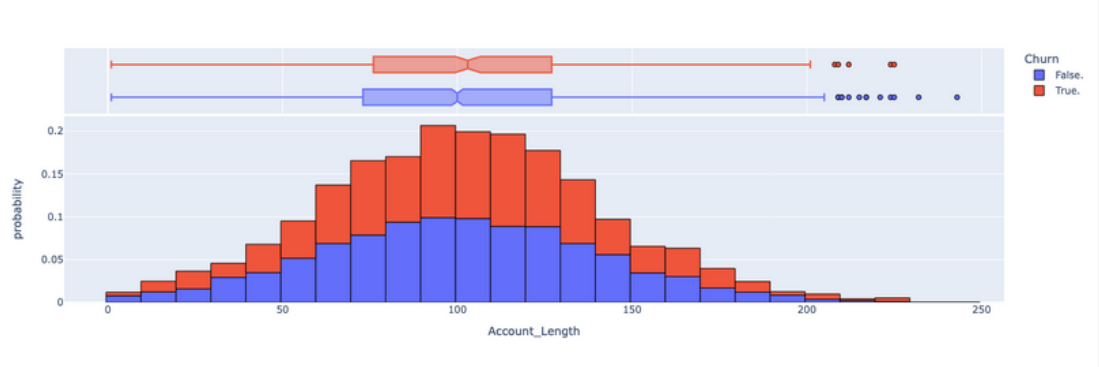
creer une fonction pour preparer nos données, on y insert toutes les applications de nettoyages que l'ont veut.

```
Ici on renomme les colonnes pour plus de clarté et pour facilité le code (on insert des "_" pour lier 2 mots et éviter les espaces)
.columns = ['State', 'Account_Length', 'Area_Code', 'Phone', 'Intl_Plan',
            'VMail_Plan', 'VMail_Message', 'Day_Mins', 'Day_Calls',
            'Day_Charge', 'Eve_Mins', 'Eve_Calls', 'Eve_Charge',
            'Night_Mins', 'Night_Calls', 'Night_Charge', 'Intl_Mins',
            'Intl_Calls', 'Intl_Charge', 'CustServ_Calls', 'Churn']

f prepare_data(data):
  # Calcul de la proportion de répondants pour chaque État
  count_by_state = data["State"].value_counts()
  proportion_by_state = count_by_state / count_by_state.sum() * 100
  data["proportion_repondants"] = data["State"].map(proportion_by_state)

  # Calcul de la proportion de churn pour chaque État
  churn_by_state = data.groupby('State')['Churn'].value_counts().unstack()
  churn_by_state['Total'] = churn_by_state['False.'] + churn_by_state['True.']
  churn_by_state['Churn_Proportion_True'] = (churn_by_state['True.'] / churn_by_state['Total']) * 100
  churn_by_state['Churn_Proportion_False'] = (churn_by_state['False.'] / churn_by_state['Total']) * 100

  # On ajoute au dataframe
  data['proportion_churn_true'] = data['State'].map(churn_by_state['Churn_Proportion_True'])
  data['proportion_churn_false'] = data['State'].map(churn_by_state['Churn_Proportion_False'])

  # On convertis les colonnes en type categories
  cols_to_convert = ['State', 'Intl_Plan', 'VMail_Plan', 'Churn']
  for col in cols_to_convert:
      data[col] = data[col].astype('category')

  return data

ta = prepare_data(df)
.0s
```

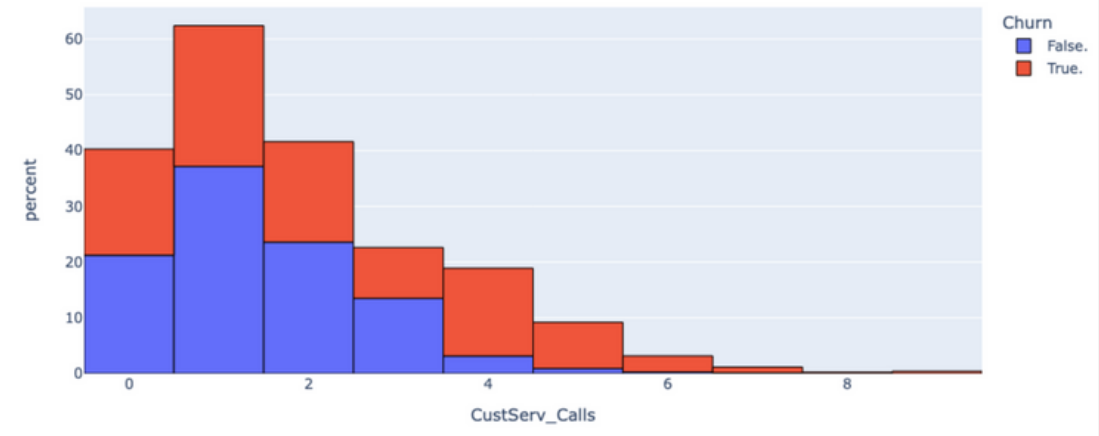
EDA (Exploratory Data Analysis)

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom

Les statistiques descriptives montrent que la durée de vie du compte client ne semble pas être un indicateur clé pour prédire le churn. En effet, les distributions de durée de vie des clients churner et non churner sont similaires, avec des moyennes et des écarts-types proches.



Nous avons remarqué qu'à partir du quatrième appel au service client, la proportion de clients churner est plus élevée que celle des clients non churner. Cette observation suggère que la satisfaction du client pourrait être un indicateur important du churn. Par conséquent, il est recommandé de surveiller de près les clients qui s'approchent de leur quatrième appel au service client afin de prévenir le churn.

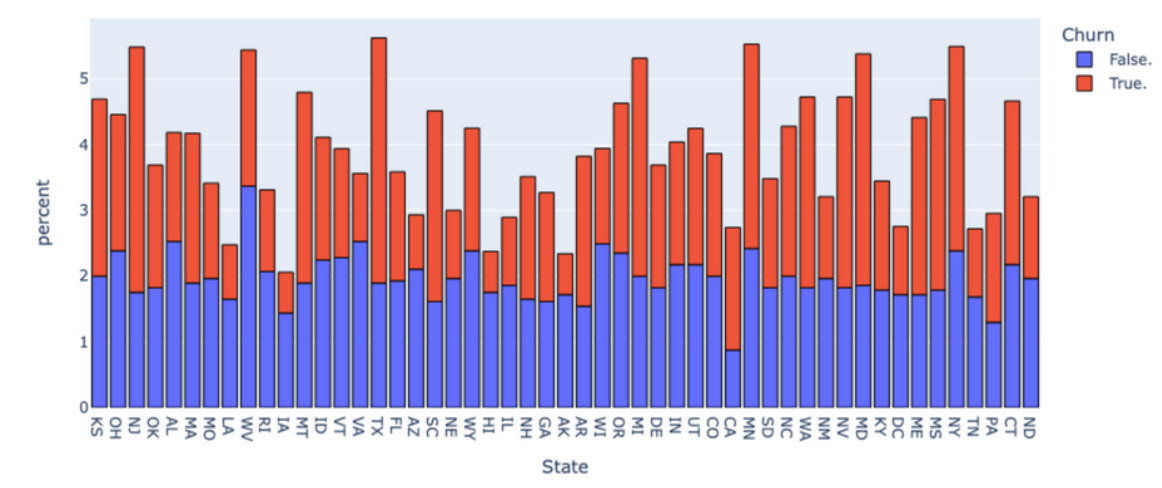



```python
# Ici on creer une boucle for pour mettre en évidence le churn par rapport a plusieur facteurs et ne pas reproduire plusieurs fois le code
for col in ['State', 'CustServ_Calls', "VMail_Plan", "Intl_Plan" ]:
    fig = px.histogram(df, x=col, color='Churn', nbins=len(df[col].unique()), histnorm='percent')
    fig.update_traces(marker=dict(line=dict(width=1, color='black')))
    fig.show()
```

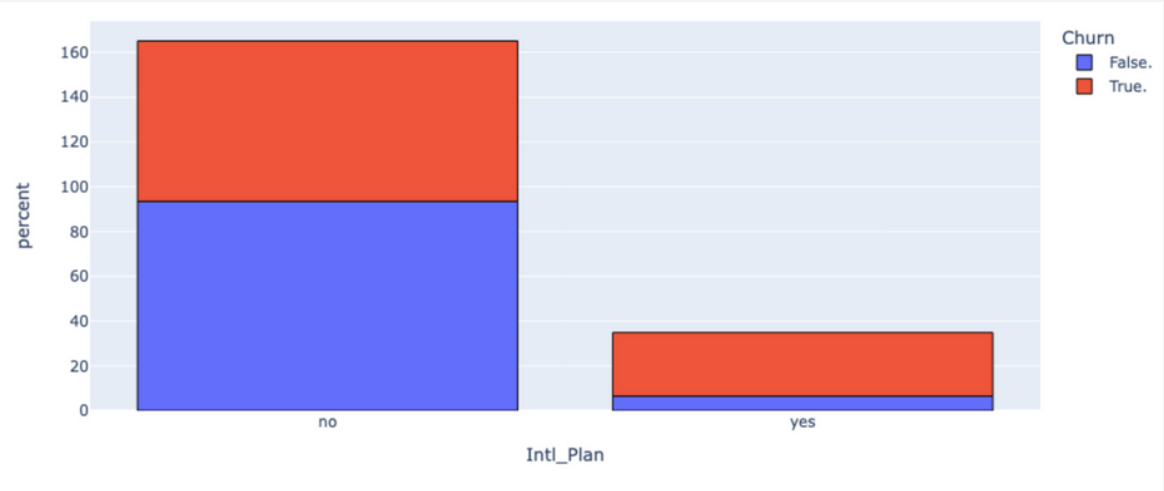
EDA (Exploratory Data Analysis)

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom

Top 5 des Etats avec le plus grand taux de Churn :
- CA
- NJ
- TX
- MD
- SC



Nous constatons qu'avec le plan international, le taux de désabonnement est beaucoup plus élevé, ce qui est une observation intéressante ! Peut-être que des dépenses importantes et mal contrôlées pour les appels internationaux sont très conflictuelles et conduisent à l'insatisfaction des clients de l'opérateur de télécommunications.



Conclusion EDA

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom

– Certains États ont un taux de désabonnement plus élevé que d'autres, ce qui pourrait être dû à des problèmes de réseau. Cependant, si un concurrent avait offert des tarifs plus bas, la plupart des États auraient un taux de désabonnement similaire.

– La zone et la durée de vie du compte n'ont aucun lien avec le taux de désabonnement et peuvent être considérées comme des données redondantes.

– Les clients avec des plans internationaux ont un taux de désabonnement plus élevé, probablement en raison de tarifs élevés ou de la couverture réseau.

– Les clients avec le plus grand nombre de minutes d'appel quotidiennes ont un taux de désabonnement plus élevé, peut-être parce qu'ils ont trouvé une entreprise proposant des tarifs plus bas.

– Aucune relation n'a été trouvée entre le taux de désabonnement et d'autres variables telles que les appels du soir ou de la nuit.

– Le taux de désabonnement augmente avec le nombre d'appels au service client. Les clients ayant appelé trois fois ou moins ont un taux de désabonnement plus faible que ceux qui ont appelé quatre fois ou plus, indiquant qu'ils avaient moins de problèmes à résoudre.

Recommandations :

– L'entreprise devrait améliorer la couverture de son réseau et résoudre les problèmes de connexion (aussi bien nationaux qu'internationaux).

– La société pourrait proposer des réductions ou créer un forfait dans lequel, lorsque le client dépasse un certain nombre de minutes d'appels par jour, il bénéficie d'une réduction sur le tarif (c'est-à-dire le coût à la minute).

– Réduire le tarif du forfait international ou proposer aux clients des remises ou des offres spéciales.

– L'entreprise pourrait améliorer son service client en apportant de meilleures solutions aux problèmes rencontrés, en recueillant les avis des clients et en travaillant sur les suggestions qui leur sont faites.

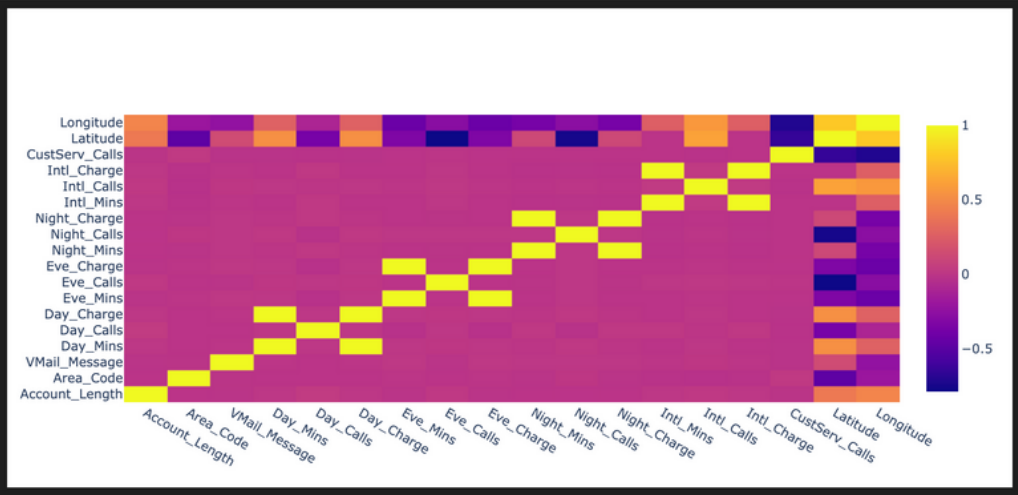
Modelisation et Prédiction

Rigaud Eddy | Rapport du Churn sur le Dataset Telecom

Matrice de corrélations des variable numériques

```
correlations = df.corr()
fig = go.Figure(go.Heatmap(z=correlations, x=correlations.columns, y=correlations.columns))
fig.show()
```

[15]



On mets en évidence laa corrélation entre les variable de Minutes et de Charges dans notre dataset.

La matrice de corrélation est un outil très utile pour comprendre les relations entre les variables d'un dataset. Elle permet de visualiser graphiquement les associations entre chaque paire de variables et de mesurer la force de ces associations.
Dans le cas de votre étude sur le churn des clients d'une entreprise de téléphonie, on observe une forte corrélation entre les variables de charges et de minutes.

Prédiction :

```
# Préparation des données pour l'entraînement et l'évaluation
X = df.drop(['Churn', 'Phone', 'State'], axis=1)
X = pd.get_dummies(X, drop_first=True)
X = X.fillna(X.mean())
y = df['Churn'].cat.codes
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[21] ✓ 0.0s

```
# Entraînement et évaluation des modèles
models = {'Random Forest': RandomForestClassifier(random_state=42),
          'Decision Tree': DecisionTreeClassifier(random_state=42),
          'SVM': SVC(random_state=42),
          'XGBoost': xgb.XGBClassifier(random_state=42)}

df_results = pd.DataFrame() # DataFrame pour stocker les prédictions
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred)
    print(f"Modèle : {model_name}")
    print(f"Rapport de classification : \n{report}")
    cm = confusion_matrix(y_test, y_pred)
    print(f"Matrice de confusion : \n{cm}\n")

    df_results[model_name] = y_pred # Ajouter une colonne pour les prédictions de chaque modèle

df_results.index = X_test.index # Définir l'index du DataFrame avec l'index de X_test
df_results['Churn'] = y_test.values # Ajouter la colonne de vraies valeurs de Churn
df_results['Phone'] = df.loc[X_test.index, 'Phone'].values # Ajouter la colonne Phone à partir du dataframe d'origine
df_results['State'] = df.loc[X_test.index, 'State'].values # Ajouter la colonne State à partir du dataframe d'origine
```

[22] ✓ 0.9s

Modelisation et Prédiction

- Modèle **Random Forest** :

Le modèle a une **précision pondérée de 95%**, avec un r**appel de 99%** pour les clients fidèles et de **69%** pour les clients churn.
Le **F1-score pondéré est de 94%**, reflétant un bon équilibre entre la précision et le rappel.
La matrice de confusion montre que le modèle a correctement classée 562 vrais négatifs et 70 vrais positifs, mais qu'il a mal classé 4 faux négatifs et 31 faux positifs.

```
Modèle : Random Forest
Rapport de classification :
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       566
           1       0.95      0.69      0.80       101

    accuracy                           0.95       667
   macro avg       0.95      0.84      0.88       667
weighted avg       0.95      0.95      0.94       667

Matrice de confusion :
[[562   4]
 [ 31  70]]
```

- Modèle **Decision Tree :**

Le modèle a une **précision pondérée de 91%,** avec un **rappel de 94%** pour les clients fidèles et de **74%** pour les clients churn.
Le **F1-score pondéré est de 91%,** reflétant un équilibre correct entre la précision et le rappel.
La matrice de confusion indique que le modèle a correctement classé 533 vrais négatifs et 75 vrais positifs, mais qu'il a mal classé 33 faux négatifs et 26 faux positifs.

```
Modèle : Decision Tree
Rapport de classification :
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       566
           1       0.69      0.74      0.72       101

    accuracy                           0.91       667
   macro avg       0.82      0.84      0.83       667
weighted avg       0.91      0.91      0.91       667

Matrice de confusion :
[[533  33]
 [ 26  75]]
```

Modelisation et Prédiction

- Modèle **SVM** :

Le modèle a une **précision pondérée de 85%,** avec un **rappel de 100%** pour les clients fidèles et de **0%** pour les clients churn.
Le F1-score pondéré est de 78%, reflétant un équilibre moins satisfaisant entre la précision et le rappel comparé aux autres modèles.
La matrice de confusion montre que le modèle a correctement classée 566 vrais négatifs, mais n'a détecté aucun vrai positif. Il y a eu 101 faux négatifs et aucun faux positif.

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

           0       0.85      1.00      0.92       566
           1       0.00      0.00      0.00       101

    accuracy                           0.85       667
   macro avg       0.42      0.50      0.46       667
weighted avg       0.72      0.85      0.78       667

Matrice de confusion :
[[566   0]
 [101   0]]
```

- Modèle **XGBoost** :

Le modèle a une **précision pondérée de 96%,** avec un **rappel de 99%** pour les clients fidèles et de **79%** pour les clients churn.
Le **F1-score pondéré est de 96%**, reflétant un excellent équilibre entre la précision et le rappel.
La matrice de confusion indique que le modèle a correctement classé 561 vrais négatifs et 80 vrais positifs, mais qu'il a mal classé 5 faux négatifs et 21 faux positifs.

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

           0       0.96      0.99      0.98       566
           1       0.94      0.79      0.86       101

    accuracy                           0.96       667
   macro avg       0.95      0.89      0.92       667
weighted avg       0.96      0.96      0.96       667

Matrice de confusion :
[[561   5]
 [ 21  80]]
```

Conclusion

En conclusion,

le modèle XGBoost est le plus performant parmi les quatre modèles étudiés, avec une précision, un rappel et un F1-score pondérés de 96%. Cela démontre qu'il est le plus efficace pour prédire correctement les clients qui vont se désabonner (churn) et ceux qui vont rester. À l'inverse, le modèle SVM est le moins performant, avec des scores nettement inférieurs aux autres modèles testés.

Les données constituent un atout indéniable pour le marketing, car elles permettent d'affiner les stratégies en se basant sur des données factuelles plutôt que sur des ressentis, comme cela était souvent le cas par le passé. Les modèles de prédiction sont également très utiles dans les campagnes marketing, parce qu'ils permettent de cibler efficacement les points faibles et de maximiser les points forts actuels.

Comme nous avons pu le constater dans cette étude de cas, le fait qu'un client quitte ou non son opérateur ne dépend pas uniquement de l'image de marque de l'entreprise. De nombreux autres facteurs sont en jeu et le marketing ne peut pas les déterminer en se basant seulement sur des ressentis. Il est donc nécessaire de se baser sur des données et des faits factuels pour élaborer une stratégie marketing efficace.