

RAPPORT DEMI-PARCOURS

CHURN PREDICTION

4DS5 2020/2021

La prédiction du churn est l'un des cas d'utilisation de Big Data les plus populaires en entreprise. Il consiste à détecter les clients susceptibles d'annuler un abonnement à un service. Bien qu'à l'origine un géant des télécommunications, cela concerne les entreprises de toutes tailles, y compris les startups.

TRAVAIL ÉLABORÉ PAR

DERBEL WASSIM PRESIDENT
HEDHILI HOURIA
BEN HASSINE OUSSAMA
NASR AZIZA
BABBA FAROUK

Table de matière

| | |
|--------------------------------------------------------------------------------------------------------------------------------|----|
| Remerciements | 5 |
| Pourquoi la Méthodologie CRISP-DM ? | 6 |
| I. Compréhension du problème métier: | 7 |
| II. Compréhension des données: | 8 |
| Analyse de forme:..... | 8 |
| Analyse de Fond:..... | 9 |
| <i>Visualisation de la variable cible :</i> | 9 |
| III. Préparation des données: | 15 |
| 1. Elimination de variables inutiles :..... | 15 |
| 2. Conversion de la variable TotalCharges en float : | 15 |
| 3. Encodage : | 16 |
| 4. Normalisation : | 17 |
| 5. Sélection des variables : | 17 |
| 5.1 Selection des variables avec la matrice de corrélation:..... | 17 |
| 5.2 Selection des variables avec Recursive Feature Elimination (RFE):..... | 18 |
| 5.3 Feature selection SelectKBest : | 18 |
| 5.4 Feature selection en utilisant variance : | 19 |
| 5.5 Feature selection en utilisant corrélation entre variables : | 20 |
| 5.6 Feature selection en utilisant la corrélation des variables avec la cible + la corrélation des variables entre eux : 20 | |
| IV. Modélisation: | 21 |
| 1. Application des modeles en utilisant la variance des variables:..... | 22 |
| 1.1 Modele KNeighbors Classifier (KNN) | 22 |
| 1.2 Modele DecisionTreeClassifier(DT)..... | 23 |
| 1.3 Modele NaiveBayes (NB) | 25 |
| 1.4 Modele LogisticRegression (LR) | 27 |
| 1.5 Modele RandomForestClassifier (RFC)..... | 28 |
| 1.6 Modele XGboost(xgb) | 29 |
| 1.7 Modele Support Vector Machine (SVM)..... | 30 |
| 1.8 Table de résumé :..... | 31 |
| 2 Application des modeles en utilisant SelectKBest | 32 |
| 2.1 Modele KNeighbors Classifier (KNN) | 32 |
| 2.2 Modele DecisionTreeClassifier(DT)..... | 33 |
| 2.3 Modele NaiveBayes (NB) | 36 |
| 2.4 Modele LogisticRegression (LR) | 38 |
| 2.5 Modele RandomForestClassifier (RFC)..... | 39 |

| | | |
|-------|---------------------------------------------------------------------------------------------------------------|----|
| 2.6 | Modele XGboost(xgb) | 40 |
| 2.7 | Modele Support Vector Machine (SVM) | 42 |
| 2.8 | Table de résumé : | 43 |
| 3 | Application des modeles en utilisant Recursive Feature Elimination (RFE) | 43 |
| 3.1 | Modele DecisionTreeClassifier(DT) | 43 |
| 3.2 | Modele LogisticRegression (LR) | 46 |
| 3.3 | Modele RandomForestClassifier (RFC) | 48 |
| 3.4 | Modele XGboost(xgb) | 50 |
| 3.5 | Table de résumé : | 51 |
| 4 | IV Application des modeles en utilisant la corrélation avec la variable cible | 52 |
| 4.1 | Modele KNeighbors Classifier (KNN) | 52 |
| 4.2 | Modele DecisionTreeClassifier(DT) | 53 |
| 4.2.5 | Text d'export du DT | 54 |
| 4.3 | Modele NaiveBayes (NB) | 55 |
| 4.3.3 | Courbe Roc du NB | 57 |
| 4.4 | Modele LogisticRegression (LR) | 57 |
| 4.5 | Modele RandomForestClassifier (RFC) | 58 |
| 4.6 | Modele XGboost(xgb) | 59 |
| 4.7 | Modele Support Vector Machine (SVM) | 61 |
| 4.8 | Table de résumé : | 62 |
| 5 | Application des modeles en utilisant la corrélation entre les variables | 62 |
| 5.1 | Modele KNeighbors Classifier (KNN) | 62 |
| 5.2 | Modele DecisionTreeClassifier(DT) | 64 |
| 5.3 | Modele NaiveBayes (NB) | 66 |
| 5.4 | Modele LogisticRegression (LR) | 68 |
| 5.4.4 | Modele RandomForestClassifier (RFC) | 69 |
| 5.4.2 | Rapport de classification du RFC | 70 |
| 5.5 | Modele XGboost(xgb) | 70 |
| 5.6 | Modele Support Vector Machine (SVM) | 72 |
| 5.7 | Table de résumé : | 72 |
| 5 | Application des modeles en utilisant la corrélation avec la variable cible + entre les variables | 73 |
| 5.1 | Modele KNeighbors Classifier (KNN) | 73 |
| 5.2 | Modele DecisionTreeClassifier(DT) | 74 |
| 5.2.3 | Courbe Roc du DT | 75 |
| 5.2.4 | Arbre de Decision du DT | 76 |
| 5.3 | Modele NaiveBayes (NB) | 77 |
| 5.4 | Modele LogisticRegression (LR) | 79 |
| 5.5 | Modele RandomForestClassifier (RFC) | 80 |
| 5.6 | Modele XGboost(xgb) | 81 |
| 5.7 | Modele Support Vector Machine (SVM) | 83 |
| 5.8 | Table de résumé : | 84 |

| | |
|-------------------------------------------------------|----|
| V. Evaluation: | 84 |
| VI. Déploiement : | 86 |
| 1. Configuration d'un environnement virtuel..... | 87 |
| 2. Installation de Django | 87 |
| 3. Création et configuration d'un projet Django | 88 |
| 4. Implementation du code de notre application | 89 |
| 5. Test de notre application..... | 90 |
| VII. Conclusion: | 93 |

Remerciements

Avant tout développement sur cette expérience, il apparaît opportun de commencer ce rapport de projet par des remerciements à ceux qui nous ont assisté au cours de ce projet, et même à ceux qui ont eu l'amabilité de faire de ce projet à un moment peu propice.

Aussi, on remercie Monsieur **Ncib Lotfi** encadrant de projet qui nous a formé et accompagné tout au long de cette expérience avec beaucoup de patience et pédagogie.

Nous remercions aussi les membres du jury d'avoir accepté d'évaluer notre travail.

On profite de cette tribune pour remercier les personnes qui de passage, ont pu nous apporter leur contribution, que ce soit au niveau des idées jusqu'à celui des conceptions, qu'elles trouvent ici l'expression de nos sincères reconnaissances.

Pourquoi la Méthodologie CRISP-DM ?

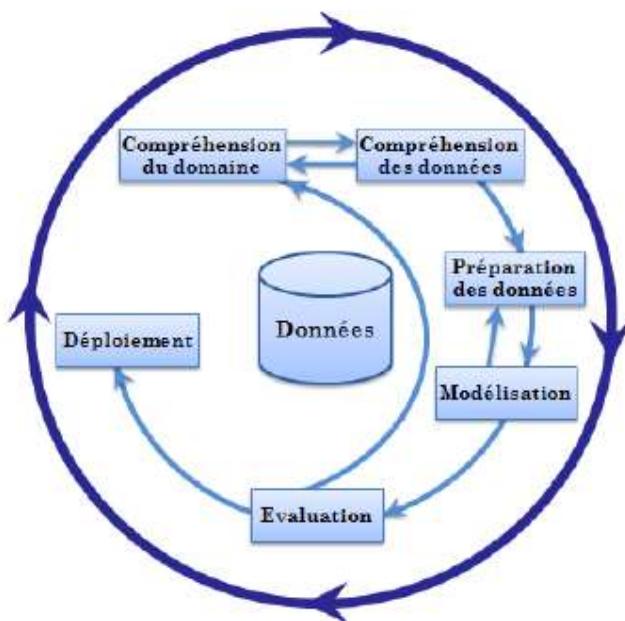
La CRISP-DM est toujours la méthodologie la plus populaire pour les analyses, le **machine learning** et les projets de sciences de données selon le blog KDnuggets. Cependant la CRISP-DM ne répond pas à toutes les contraintes des projets data science et une autre alternative est attendue depuis longtemps.

La méthode CRISP (initialement connue comme CRISP-DM) a été au départ développée par IBM dans les années 60 pour réaliser les projets Datamining. Elle reste aujourd’hui la seule méthode utilisable efficacement pour tous les projets Data Science.

Les 6 étapes de la méthodologie CRISP-DM représentent toujours une bonne description pour le processus d’analyse, par contre les détails et les besoins spécifiques doivent être mis à jour. Concrètement, **CRISP-DM** n'est pas adapté aux défis actuels du **Big Data** et de la science des données, parce que nouveaux besoin sont apparus comme le stream processing et le real time processing.

Les data scientists reconnaissent qu'une collaboration avec des partenaires métier est essentielle pour offrir une valeur business. Ils ont également reconnu que de nombreux modèles d'analyse ne sont pas déployés ou ne fournissent pas la valeur business attendue. Les data scientists adoptent ce qu'on appelle la modélisation des décisions pour remédier à ces lacunes.

Elle reste aujourd’hui (la seule?!) méthode utilisable efficacement pour tous les projets Data Science. Cette méthode est agile et itérative, c'est-à-dire que chaque itération apporte de la connaissance métier supplémentaire qui permet de mieux aborder l’itération suivante. Elle offre un aperçu du cycle de vie de l’exploration des données.



I. Compréhension du problème métier:



Le churn est le phénomène de perte de client constitue une vraie problématique pour les organisations dans différents secteurs d'activité y compris la télécommunication.

L'étude sur le Churn a récemment suscité un intérêt considérable pour les parties prenantes en raison des pertes de revenus associées. En effet Les opérateurs de télécommunications se rendent compte de l'importance de retenir les clients déjà existants au lieu d'essayer d'acquérir des nouveaux car la saturation du marché fait que presque tous les clients potentiels sont déjà dans la concurrence d'autant plus que le coût engagé pour ajouter un nouveau client est beaucoup plus important que de retenir un client dont l'appétit n'est pas correctement servi.

Les clients sont l'un des atouts les plus importants d'une entreprise dynamique et compétitive . Slogan «Le client a toujours raison» qui exhorte l'entreprise à donner une priorité élevée et le meilleur service à la satisfaction du client. Ainsi, une stratégie globale de développement, de gestion et de renforcement des clients pérennes une prévision de désabonnement client utilisant Pearson la relation relève de l'équipe de gestion de la relation client L'un des défis pour la société de télécommunications est de maintenir la fidélité du client. En effet, la perte de clients pourrait entraîner une perte de revenu critique. Comme le prévoit fidéliser l'ancien client est cinq à six fois moins cher que de trouver un nouveau client. Par conséquent, il est nécessaire pour la société de télécommunications d'avoir la capacité de prédire le client qui lui sera fidèle sans action d'intervention qui pourrait entraîner une perte de revenu, des dépenses supplémentaires de fidélisation et de réacquisition de la clientèle, des coûts publicitaires supplémentaires, un chaos organisationnel ainsi que de planification et de budgétisation. En dehors de cela, les clients qui quittent la société de télécommunications pourraient inciter d'autres à faire de même. Par la suite, afin de maintenir la propension ou la tendance des clients à l'égard de l'entreprise, ils doivent tenir compte du comportement des clients et fournir les meilleurs services en tenant compte des préférences des clients. C'est ce qu'on appelle la «prédiction du taux de désabonnement des clients». Il est essentiel de mettre en œuvre la prédiction du taux de désabonnement dans leur approche de prévision des clients à haut risque.

L'apprentissage automatique pourrait être le genre d'outils qui pourraient aider les entreprises de télécommunications dans le domaine de la prédiction du taux de désabonnement. L'apprentissage automatique fait partie de l'intelligence artificielle qui permet à l'ordinateur d'apprendre l'algorithme automatiquement sans intervention humaine. Les tâches d'apprentissage automatique telles que la classification permettent aux ordinateurs d'utiliser les données existantes pour prévoir les comportements, les résultats et les tendances futurs. L'étiquette de classification dans le modèle de prédiction du taux de désabonnement est client de désabonnement et non-désabonnement. Il existe des études sur la prédiction du taux de désabonnement à l'aide d'algorithmes d'apprentissage automatique La plupart des études utilisent des apprentissages supervisés tels que les arbres de décision, la machine à vecteurs de support (SVM) et Naïve Bayes. KneigborsClassifier (KNN)

II. Compréhension des données:

Après la collecte des données, plusieurs étapes sont effectuées pour explorer les données. L'objectif de cette étape est de comprendre ces données.

Analyse de forme:

Variable cible :

Notre variable libellée est la variable **Churn** qui présente deux classes : (1 / 0)

1 : Le client a quitté le service.

0 : Le client consomme encore le service.

Lignes et colonnes :

7043 lignes => 7043 observations

33 colonnes => 33 caractéristiques

Types de variables :

```
dtypes: float64(3), int64(6), object(24)
memory usage: 1.8+ MB
```

➔ 24 qualitatives/9 quantitatives

En visualisant le type de chaque variable on constate que la variable **TotalCharges** est de type Object (on attend à l'avoir de type float)

| | | | | |
|----|-------------------|------|----------|---------|
| 0 | CustomerID | 7043 | non-null | object |
| 1 | Count | 7043 | non-null | int64 |
| 2 | Country | 7043 | non-null | object |
| 3 | State | 7043 | non-null | object |
| 4 | City | 7043 | non-null | object |
| 5 | Zip Code | 7043 | non-null | int64 |
| 6 | Lat Long | 7043 | non-null | object |
| 7 | Latitude | 7043 | non-null | float64 |
| 8 | Longitude | 7043 | non-null | float64 |
| 9 | Gender | 7043 | non-null | object |
| 10 | Senior Citizen | 7043 | non-null | object |
| 11 | Partner | 7043 | non-null | object |
| 12 | Dependents | 7043 | non-null | object |
| 13 | Tenure Months | 7043 | non-null | int64 |
| 14 | Phone Service | 7043 | non-null | object |
| 15 | Multiple Lines | 7043 | non-null | object |
| 16 | Internet Service | 7043 | non-null | object |
| 17 | Online Security | 7043 | non-null | object |
| 18 | Online Backup | 7043 | non-null | object |
| 19 | Device Protection | 7043 | non-null | object |
| 20 | Tech Support | 7043 | non-null | object |
| 21 | Streaming TV | 7043 | non-null | object |
| 22 | Streaming Movies | 7043 | non-null | object |
| 23 | Contract | 7043 | non-null | object |
| 24 | Paperless Billing | 7043 | non-null | object |
| 25 | Payment Method | 7043 | non-null | object |
| 26 | Monthly Charges | 7043 | non-null | float64 |
| 27 | Total Charges | 7043 | non-null | object |
| 28 | Churn Label | 7043 | non-null | object |
| 29 | Churn Value | 7043 | non-null | int64 |
| 30 | Churn Score | 7043 | non-null | int64 |
| 31 | CLTV | 7043 | non-null | int64 |
| 32 | Churn Reason | 1869 | non-null | object |

Figure 1 : Type des données

Visualisation de la variable TotalCharges :

```
array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
      dtype=object)
```

Figure 2 : Visualisation de la variable Totalcharges

➔ On comprend qu'il faut convertir cette variable en float dans l'étape de la préparation de données.

Analyse de valeurs manquantes :

```
Out[77]: CustomerID          0.000000
Count              0.000000
Country            0.000000
State              0.000000
City               0.000000
Zip Code            0.000000
Lat Long            0.000000
Latitude           0.000000
Longitude           0.000000
Gender              0.000000
Senior citizen      0.000000
Partner             0.000000
Dependents          0.000000
Tenure Months       0.000000
Phone Service        0.000000
Multiple Lines       0.000000
Internet Service     0.000000
Online Security       0.000000
Online Backup          0.000000
Device Protection     0.000000
Tech Support          0.000000
Streaming TV          0.000000
Streaming Movies        0.000000
Contract             0.000000
Paperless Billing      0.000000
Payment Method         0.000000
Monthly Charges        0.000000
Total Charges          0.000000
Churn Label           0.000000
Churn Value            0.000000
Churn Score            0.000000
CLTV                  0.000000
Churn Reason           73.463013
dtype: float64
```

Figure 3 : Présence des valeurs Nan

- ➔ Présence des valeurs manquantes

Analyse de Fond:

Visualisation de la variable cible :

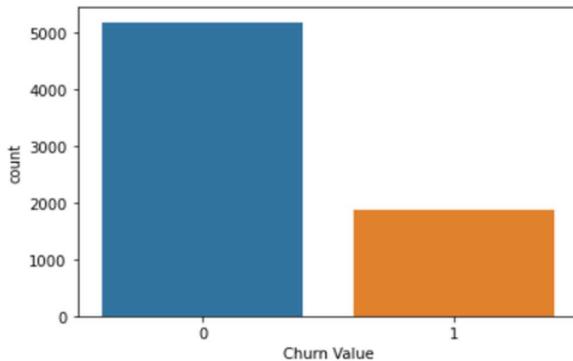


Figure 4 : Pourcentage de Churn

- ➔ Dans nos données, 5000 des clients n'abandonnent pas.
- ➔ Dataset non équilibrée

Relation des variables avec la variable cible :

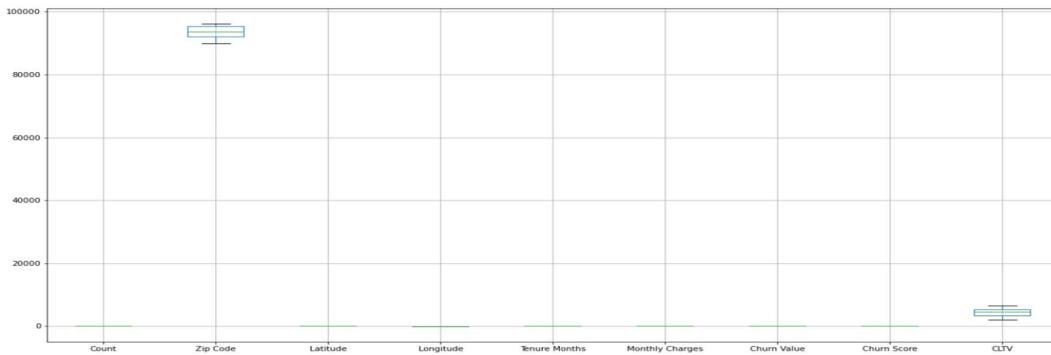


Figure 5 : les Boxplot pour toutes les variables

- ➔ la variable 'Zip Code' dispose des valeurs très grandes qui cause une perte d'information , on va supprimer cette variable dans la phase préparation des données.

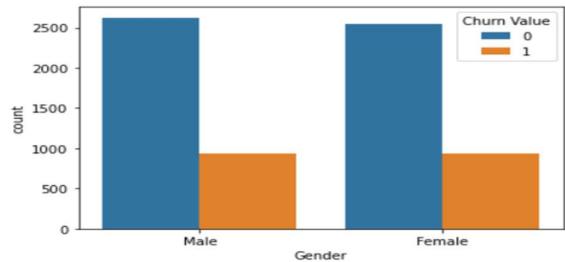


Figure 6 : Distribution par gender

- ➔ on peut conclure que la distribution est équilibrée
- ➔ donc la variable '**Gender**' ne joue aucun rôle dans notre prédition

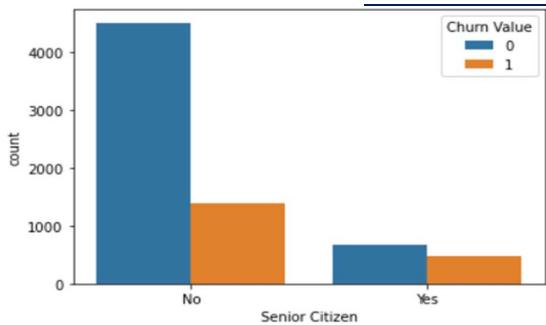


Figure 7 : Pourcentage des jeunes citoyens

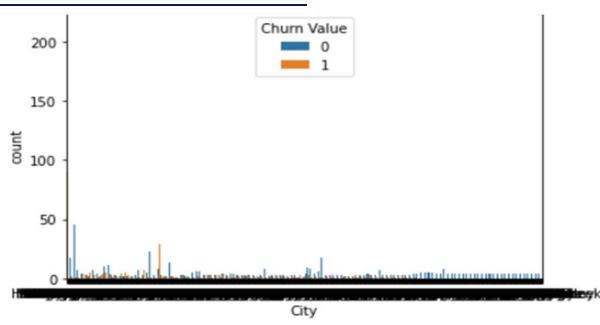


Figure 8 : Effet de variable city sur churn value

- ➔ on remarque que la plupart de nos clients dans les données sont des personnes plus jeunes
- ➔ on remarque la variable 'City' n 'apporte aucune information ' sur la variable '**Churn Value**'

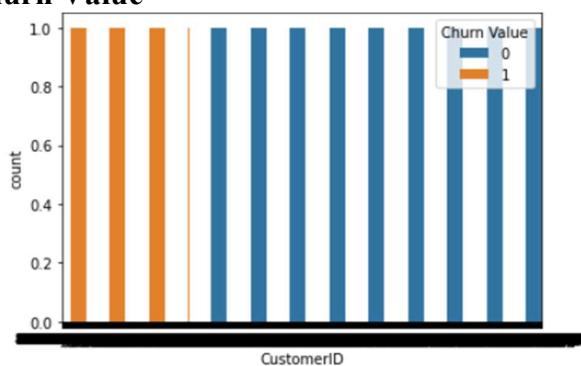


Figure 9 : Effet CustomerID

- ➔ On remarque la variable '**CustomrerID**' n 'apporte aucune information ' sur la variable '**Churn Value**'.

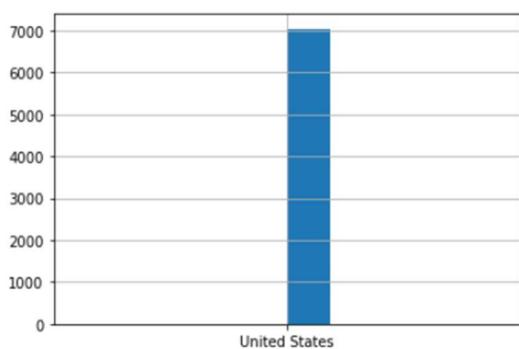


Figure 10 : Effet de la variable Country

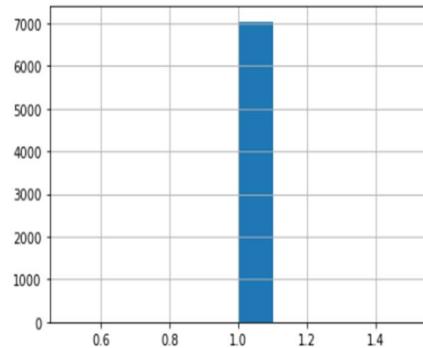


Figure 11 : Effet de la variable Count

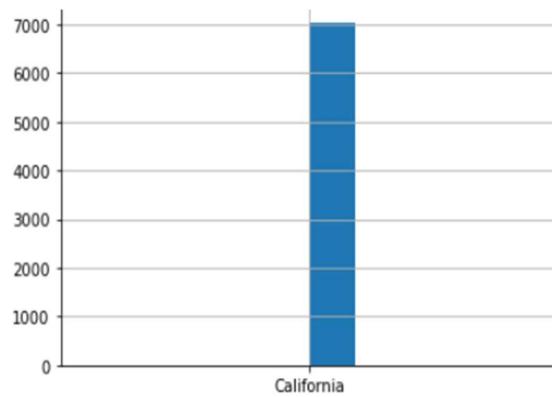


Figure 12 : Effet de la variable State

- ➔ On remarque les variables '**State**', '**Count**', '**Country**' n'apportent aucune information sur la variable '**Churn Value**'.
- ➔ On va supprimer cette variable dans la phase préparation des données.

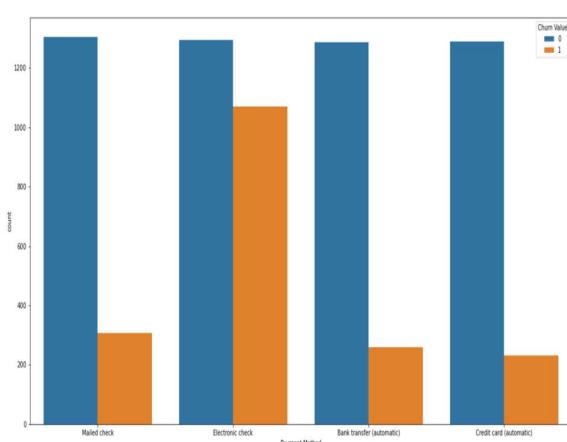


Figure 13 : Effet de la variable Payment Method

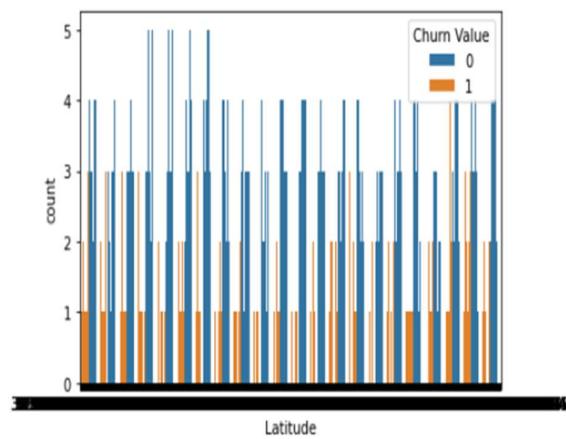


Figure 14: Effet de la variable Latitude

- ➔ On remarque que les clients qui suivent la méthode de paiement : ' Electronic check' ont un risque de quitte ->La variable '**Latitude**' n'apporte aucune information de notre prédiction

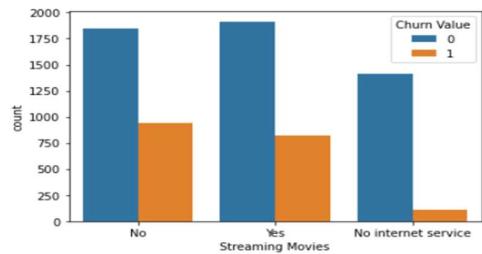


Figure 15: Effet de la variable Streaming Movies

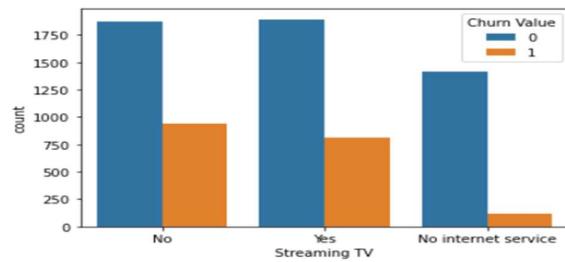


Figure 16: Effet de la variable Streaming TV

→ Les deux variables 'Streaming Movies' et 'Streaming TV' apportent la même information

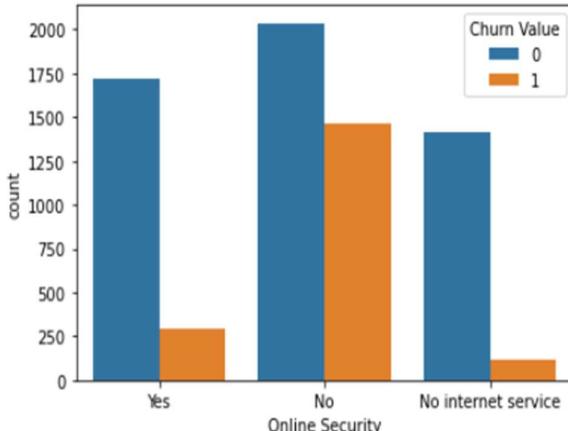


Figure 17: Effet de la variable Online Security

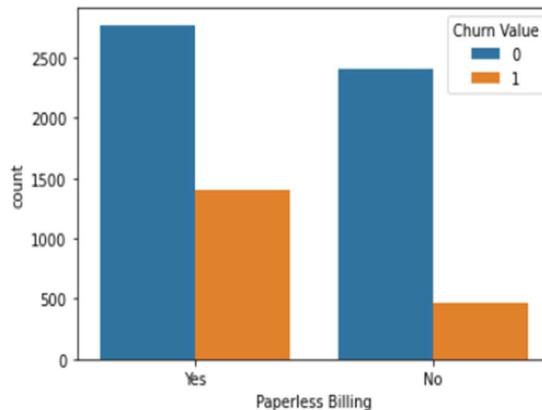


Figure 18: Effet de la variable Paperless Billing

→ Une grande pourcentage des clients qui ont quitté n'ont pas une **Online Security**

->Une grande pourcentage des clients qui ont quitté ont une **Parpeless Billing**

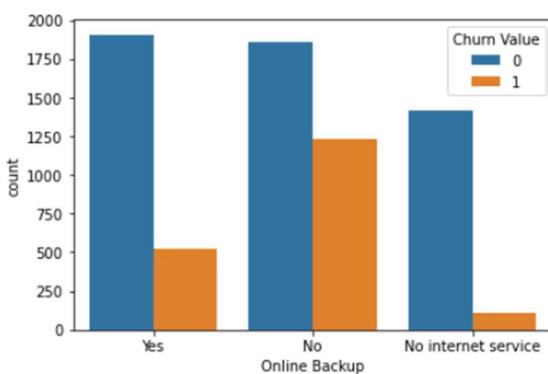


Figure 19: Effet de la variable Online Backup

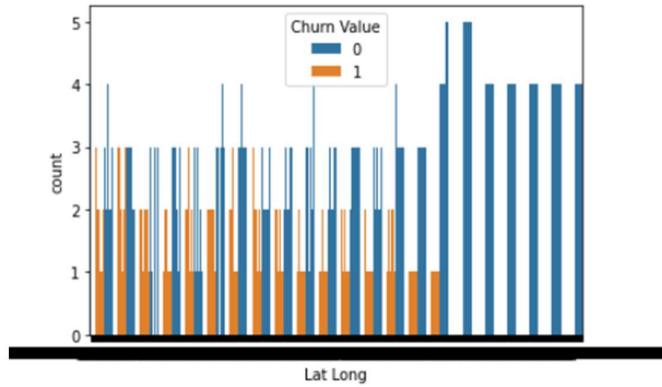


Figure 20: Effet de la variable Lat Long

→ Une grande pourcentage des clients qui ont quitté n'ont pas une **Online Backup**.

->La variable '**LatLong**' n'apporte aucune information pour notre prédiction

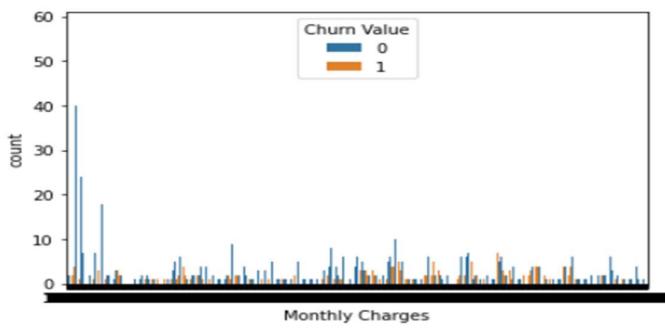


Figure 21: Effet de la variable Monthly Charges

→ La variable ' Monthly Charges ' peut nous aider dans notre prédiction.garder cette variable.

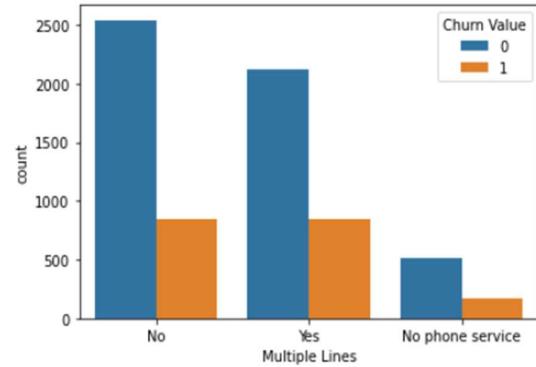


Figure 22: Effet de la variable Multiple Lines

->on ne peut pas conclure on va aider dans

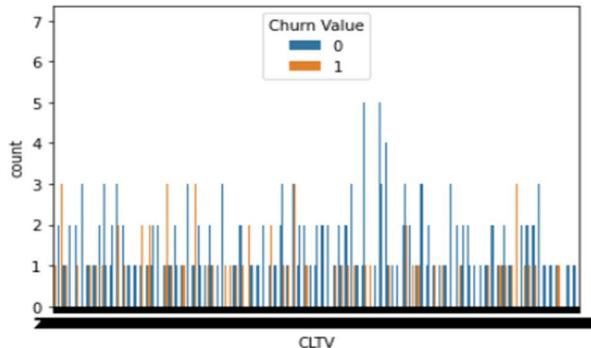


Figure 23: Effet de la variable CTV

→ La variable ' CTV ' n'apporte aucune information pour notre prédiction.

-> Une grande pourcentage des clients qui ont un Fibre Optique ont quittés

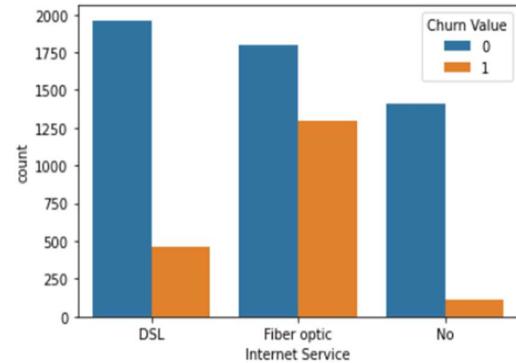


Figure 24: Effet de la variable Internet Service

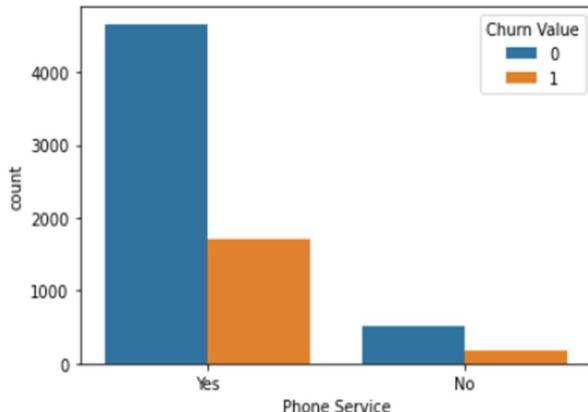


Figure 26: Effet de la variable Phone Service

→ Une grande pourcentage des clients qui ont quitté ont un Phone Service.

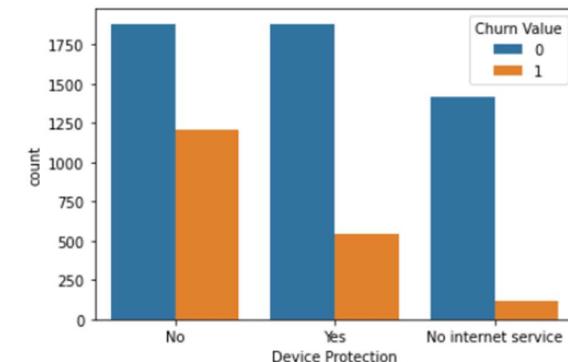


Figure 27: de la variable Device Protection

->Une grande pourcentage des clients qui ont quitté n'ont pas une Device Protection

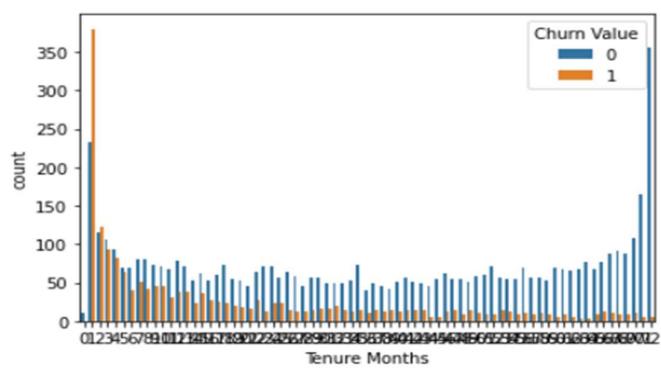


Figure 28: Effet de la variable Tenure Months

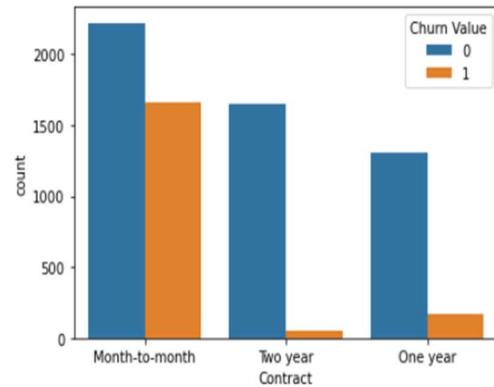


Figure 29: Effet de la variable Contract

- ➔ On ne peut pas interpréter on garde La variable 'Tenure Months' pour notre prédiction.mensuel. Alors qu'il y a un nombre égal de clients dans les contrats de 1 an et 2 ans
- ➔ La plupart des clients sont dans le contrat 'Month-to-month'.

III. Préparation des données:

1. Elimination de variables inutiles :

Cette étape consiste à visualiser les colonnes de Dataset et détecter à première vue les colonnes qui sont inutiles à la classification et qui peuvent même fausser les résultats. Ces colonnes doivent être supprimées

Exemples : les colonnes à variables constantes/ les identifications des individus

Par une visualisation initiale de notre Dataset , on remarque que les colonnes 'CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code', 'Lat Long', 'Latitude', 'Longitude', 'Churn Label', 'CLTV', 'Churn Reason' n'apporte aucun effet sur le résultat de prédiction de Churn .

➔ Elimination des variables 'CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code', 'Lat Long', 'Latitude', 'Longitude', 'Churn Label', 'CLTV', 'Churn Reason'.

| | Gender | Senior Citizen | Partner | Dependents | Tenure Months | Phone Service | Multiple Lines | Internet Service | Online Security | Online Backup | ... |
|---|--------|----------------|---------|------------|---------------|---------------|----------------|------------------|-----------------|---------------|-----|
| 0 | Male | No | No | No | 2 | Yes | No | DSL | Yes | Yes | ... |
| 1 | Female | No | No | Yes | 2 | Yes | No | Fiber optic | No | No | ... |
| 2 | Female | No | No | Yes | 8 | Yes | Yes | Fiber optic | No | No | ... |
| 3 | Female | No | Yes | Yes | 28 | Yes | Yes | Fiber optic | No | No | ... |
| 4 | Male | No | No | Yes | 49 | Yes | Yes | Fiber optic | No | Yes | ... |

5 rows × 21 columns



Figure: Visualisation des colonnes

2. Conversion de la variable TotalCharges en float :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 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  TotalCharges    7032 non-null   float64 
 19  Churn           7043 non-null   object  
dtypes: float64(2), int64(2), object(16)
memory usage: 1.1+ MB
```

Figure 13 : Visualisation de type des colonnes

En contrepartie, les observations qui n'ont pas été convertis, seront remplacés par une valeur NAN

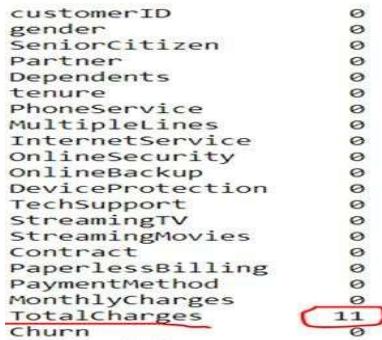


Figure 14 : Visualisation de nombre des variables Nan

- 11 valeurs NAN à partir de 7043 valeurs (11 est négligeable devant 7043)
- On décide d'éliminer les 11 valeurs manquantes par un dropna().

3. Encodage :

L'encodage consiste à convertir des données catégorielles en données numériques.

Un type de données catégorique était le type de données le plus courant présent dans l'ensemble de données.

Les algorithmes d'apprentissage automatique ne fonctionnent que sur des données numériques car ils sont basés sur des équations mathématiques. Donc, il était impossible de garder les variables catégoriques telles qu'elles étaient et il était nécessaire de convertir ces variables sous forme numérique.

Dans cette recherche, la méthode d'encodage **OneHot** a été utilisée pour coder les variables.

| Dependents_Yes | Phone_Service_No | Phone_Service_Yes | "" | Paperless_Billing_Yes | Method_Bank_transfer (automatic) | Method_Credit_card (automatic) | Payment_Method_Electronic_check | Payment_Method_Mailed_check | Tenure_Months | Monthly_Charges | Total_Charges | Churn_Score | Ch_V |
|----------------|------------------|-------------------|-----|-----------------------|-------------------------------------|-----------------------------------|---------------------------------|-----------------------------|---------------|-----------------|---------------|-------------|------|
| 0 | 0 | 1 ... | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 53.85 | 108.15 | 86 | |
| 1 | 0 | 1 ... | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 70.70 | 151.65 | 67 | |
| 1 | 0 | 1 ... | 1 | 0 | 0 | 0 | 1 | 0 | 8 | 99.65 | 820.50 | 86 | |
| 1 | 0 | 1 ... | 1 | 0 | 0 | 0 | 1 | 0 | 28 | 104.80 | 3046.05 | 84 | |
| 1 | 0 | 1 ... | 1 | 1 | 0 | 0 | 0 | 0 | 49 | 103.70 | 5036.30 | 89 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 0 | 0 | 1 ... | 1 | 1 | 0 | 0 | 0 | 0 | 72 | 21.15 | 1419.40 | 45 | |
| 1 | 0 | 1 ... | 1 | 0 | 0 | 0 | 0 | 1 | 24 | 84.80 | 1990.50 | 59 | |
| 1 | 0 | 1 ... | 1 | 0 | 1 | 0 | 0 | 0 | 72 | 103.20 | 7362.90 | 71 | |
| 1 | 1 | 0 ... | 1 | 0 | 0 | 0 | 1 | 0 | 11 | 29.60 | 346.45 | 59 | |
| 0 | 0 | 1 ... | 1 | 1 | 0 | 0 | 0 | 0 | 66 | 105.65 | 6844.50 | 38 | |

Figure : Visualisation d'encodage

4. Normalisation :

Les algorithmes d'apprentissage ont tendance à être plus performants ou à converger plus rapidement lorsque les différentes caractéristiques (variables) sont à plus petite échelle. Par conséquent, il est courant de normaliser les données avant d'y entraîner des modèles d'apprentissage automatique.

La normalisation rend également le processus d'entraînement moins sensible à l'échelle des fonctionnalités. Cela permet d'obtenir de meilleurs coefficients après l'entraînement.

La normalisation min-max est l'un des moyens les plus courants de normaliser les données. Pour chaque fonctionnalité, la valeur minimale de cette fonctionnalité est transformée en 0, la valeur maximale est transformée en 1 et toutes les autres valeurs sont transformées en un nombre décimal compris entre 0 et 1.

$$X_{\text{Normalisé}} = \frac{\text{Valeur} - \text{Min}}{\text{Max} - \text{Min}}$$

]:

| | Gender_Male | Senior_Citizen_Yes | Partner_Yes | Dependents_Yes | Phone_Service_Yes | Multiple_Lines_No_phone_service | Multiple_Lines_Yes | Internet_Service_Fiber_optic | Internet_Service_No | Online_Security_No_internet_service | ... | Contract_One_year |
|------|-------------|--------------------|-------------|----------------|-------------------|---------------------------------|--------------------|------------------------------|---------------------|-------------------------------------|-----|-------------------|
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 7038 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | ... | 0 |
| 7039 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7040 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 7041 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7042 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

7032 rows × 31 columns

Figure : Normalisation des variables

5. Sélection des variables :

Pour quoi faire la sélection des variables ?

- L'élimination des Facteurs sans influence ou peu influents.
- L'élimination des Facteurs redondants.
- Diminuer la dimension des entrées telle que coût de l'apprentissage trop grand.
- Apprentissage moins coûteux
- Faciliter l'apprentissage :

*Meilleure performance en classification

* Meilleure compréhensibilité de l'hypothèse

5.1 Selection des variables avec la matrice de corrélation:

La matrice de corrélation a été réalisée pour analyser la corrélation entre les variable indépendante et dépendante et aussi pour identifier la multicolinéarité entre les fonctionnalités indépendantes. La méthode de corrélation «Spearman» a été utilisée pour 17

identifier corrélation car la thèse se compose à la fois de variables continues et catégorielles.
La carte thermique de corrélation a été générée comme indiqué dans la figure ci-dessous

Par cette méthode on va sélectionner les variables qui sont corrélés avec la variable cible en choisissant un seuil égale 0.2

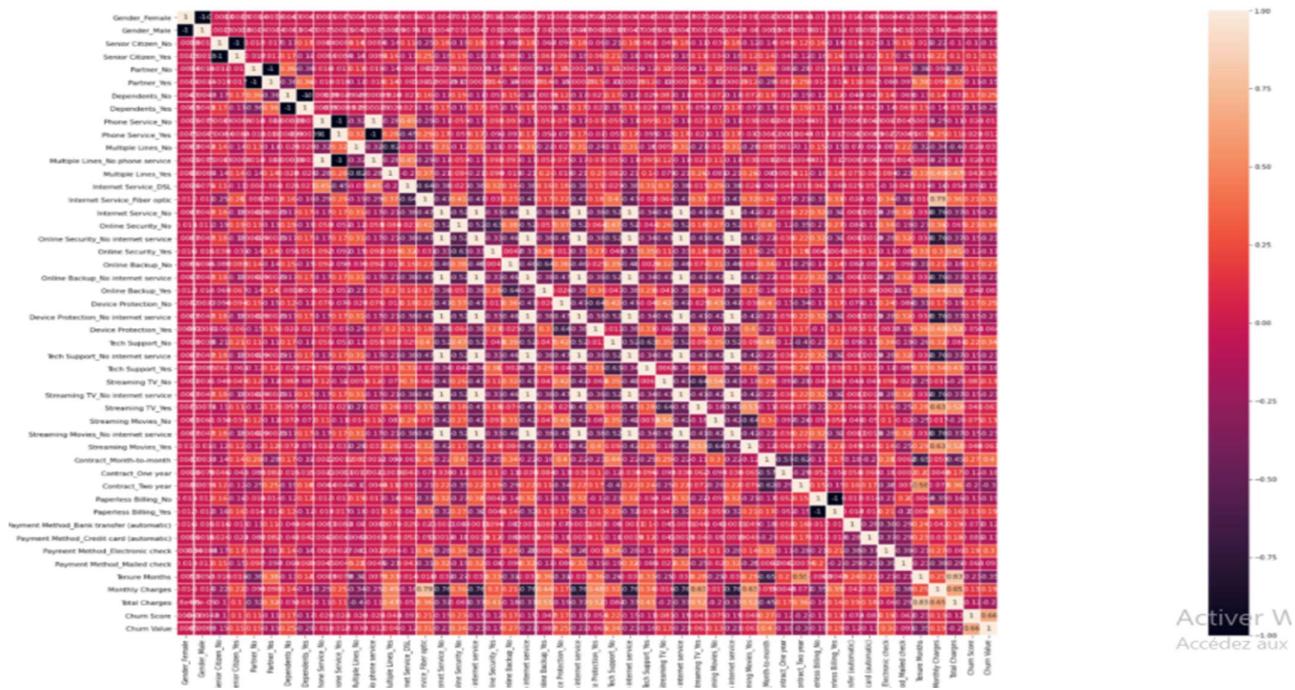


Figure : Matrice de corrélation

➔ On va travailler avec 13 variables

5.2 Selection des variables avec Recursive Feature Elimination (RFE):

En effet, RFE est une méthode de type «wrapper» qui va avoir tendance à sélectionner des variables comportant de l'information complémentaire, améliorant ainsi la tâche de classification. Les attributs considérés un à un dans la sélection ne contiennent que peu d'information pertinente.

```
Sélection de variables [ True  True  True  True  False  False  True  False  False  False
False  False  False  False  False  False  False  False  False  False  True
False  False  False  True  True  True]
Classement de variables [ 1  1  1  1 16 18  6  1 17 19  4 22  2  8 13 20  5 15 12 21 10 11 14  1
7 3 9 1 1 1]
Variables sélectionnées : ['Gender_Male', 'Senior Citizen_Yes', 'Partner_Yes', 'Dependents_Yes', 'Internet Service_Fiber optic', 'Paperless Billing_Yes', 'Tenure Months', 'Monthly Charges', 'Total Charges']
```

Figure : Feature_selection RFE

➔ On va travailler avec 9 variables

5.3 Feature selection SelectKBest :

Sélection univariée de variables (SelectKBest): est un transformer plus puissant que

VarianceThreshold. Il utilise des tests de dépendance (khi2, Information mutuelle, F-mesure) pour sélectionner un nombre K de variables ayant un lien fort avec la variable target y.

```
4]: from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

kbest = SelectKBest(score_func=f_classif, k=8)
kbest.fit(x_train, y_train)

print("Sélection de variables : ", kbest.get_support())
print("Scores de variables : ", kbest.scores_)
print("Variables sélectionnées : ", list(x.columns[kbest.get_support()]))
print("Variables supprimées : ", list(x.columns[~kbest.get_support()]))
```

Sélection de variables : [False False False True False False True False False False
 False False False True False True False True False False True False
 False True False True False False]
 Scores de variables : [1.87294194e+00 1.27700594e+02 1.45451945e+02 3.48674421e+02
 8.15083517e-01 8.15083517e-01 5.45566320e+00 5.56681760e+02
 2.90823731e+02 2.90823731e+02 1.71907013e+02 2.90823731e+02
 4.82344312e+01 2.90823731e+02 3.14735342e+01 2.90823731e+02
 1.72442439e+02 2.90823731e+02 1.80580832e+01 2.90823731e+02
 1.55931993e+01 2.07501972e+02 5.37062295e+02 2.00714457e+02
 9.77635435e+01 5.47500073e+02 4.35468330e+01 8.22122532e+02
 1.94858122e+02 2.45200246e+02]
 Variables sélectionnées : ['Dependents_Yes', 'Internet Service_Fiber optic', 'Tech Support_No internet service', 'Streaming TV_No internet service', 'Streaming Movies_No internet service', 'Contract_Two year', 'Payment Method_Electronic check', 'Tenure Months']
 Variables supprimées : ['Gender_Male', 'Senior Citizen_Yes', 'Partner_Yes', 'Phone Service_Yes', 'Multiple Lines_No phone service', 'Multiple Lines_Yes', 'Internet Service_No', 'Online Security_No internet service', 'Online Security_Yes', 'Online Backup_No internet service', 'Online Backup_Yes', 'Device Protection_No internet service', 'Device Protection_Yes', 'Tech Support_Yes', 'Streaming TV_Yes', 'Streaming Movies_Yes', 'Contract_One year', 'Paperless Billing_Yes', 'Payment Method_Credit card (automatic)', 'Payment Method_Mailed check', 'Monthly Charges', 'Total Charges']

```
5]: x_train = kbest.transform(x_train)
x_test = kbest.transform(x_test)
```

Figure : Feature_selectionSelectKBest

➔ On va travailler avec 8 variables

5.4 Feature selection en utilisant variance :

VarianceThreshold est une approche de base simple de la sélection des fonctionnalités. Il supprime toutes les fonctionnalités dont la variance ne dépasse pas un certain seuil. Par défaut, il supprime toutes les fonctionnalités à variance nulle, c'est-à-dire les fonctionnalités qui ont la même valeur dans tous les échantillons.

```
5]: # Perform feature selection using a variance threshold
from sklearn.feature_selection import VarianceThreshold

sel = VarianceThreshold(threshold=(0.1))
sel.fit(x_train)

print("Sélection de variables : ", sel.get_support())
print("Variables sélectionnées : ", list(x.columns[sel.get_support()]))
print("Variables supprimées : ", list(x.columns[~sel.get_support()]))
```

Sélection de variables : [True True True True False False True True True True
 True True True True True True True True True True True True True True
 True True True True False False]
 Variables sélectionnées : ['Gender_Male', 'Senior Citizen_Yes', 'Partner_Yes', 'Dependents_Yes', 'Multiple Lines_Yes', 'Internet Service_Fiber optic', 'Internet Service_No', 'Online Security_No internet service', 'Online Security_Yes', 'Online Backup_No internet service', 'Online Backup_Yes', 'Device Protection_No internet service', 'Device Protection_Yes', 'Tech Support_No internet service', 'Tech Support_Yes', 'Streaming TV_No internet service', 'Streaming TV_Yes', 'Streaming Movies_No internet service', 'Streaming Movies_Yes', 'Contract_One year', 'Contract_Two year', 'Paperless Billing_Yes', 'Payment Method_Credit card (automatic)', 'Payment Method_Electronic check', 'Payment Method_Mailed check', 'Tenure Months']
 Variables supprimées : ['Phone Service_Yes', 'Multiple Lines_No phone service', 'Monthly Charges', 'Total Charges']

```
7]: x_train = sel.transform(x_train)
x_test = sel.transform(x_test)
```

Figure : Feature_selection_variance

➔ On va travailler avec 26 variables

5.5 Feature selection en utilisant corrélation entre variables :

Par cette méthode on va sélectionner les variables qui sont corrélées entre eux

```
3]: def correlatedFeatures(dataset, threshold):
    correlated_columns = set()
    correlations = dataset.corr()
    for i in range(len(correlations)):
        for j in range(i):
            if abs(correlations.iloc[i,j]) > threshold:
                correlated_columns.add(correlations.columns[i])
    return correlated_columns

4]: cf = correlatedFeatures(x_train, 0.85)
cf

5]: {'Device Protection_No internet service',
      'Multiple Lines_No phone service',
      'Online Backup_No internet service',
      'Online Security_No internet service',
      'Streaming Movies_No internet service',
      'Streaming TV_No internet service',
      'Tech Support_No internet service'}

6]: x_train = x_train.drop(cf, axis=1)
x_test = x_test.drop(cf, axis=1)
print(x_train.shape)
print(x_test.shape)

(5625, 23)
(1407 23)
```

Figure : Variables corrélées entre eux

➔ On va travailler avec 23 variables

5.6 Feature selection en utilisant la corrélation des variables avec la cible + la corrélation des variables entre eux :

Par cette méthode on va sélectionner les variables qui sont corrélées avec la variable cible en choisissant un seuil égal à 0.2 après on va appliquer la méthode de corrélation des variables entre eux au seuil de 0.85 le résultat est mentionné ci dessous

```
Out[20]: {'Device Protection_No internet service',
          'Online Backup_No internet service',
          'Online Security_No internet service',
          'Streaming Movies_No internet service',
          'Streaming TV_No internet service',
          'Tech Support_No internet service'}
```

Figure : Variables corrélées avec target + variables entre eux

➔ On va travailler avec 7 variables

IV. Modélisation:

C'est la phase de Data Science proprement dite. La modélisation comprend le choix, le paramétrage et le test de différents algorithmes ainsi que leur enchaînement, qui constitue un modèle. Ce processus est d'abord descriptif pour générer de la connaissance, en expliquant pourquoi les choses se sont passées. Il devient ensuite prédictif en expliquant ce qu'il va se passer, puis prescriptif en permettant d'optimiser une situation future.

Tout d'abord on a divisé les données en données d'apprentissage et données de test

On a pris la taille de test set égale à 0.2 et la taille train set égale à 0.8

On va utiliser comme algorithmes pour prédire le taux de désabonnement des clients:

K-Nearest Neighbours (k-NN voire KNN ou méthode des k plus proches voisins) :

k-NN est un algorithme standard de classification qui repose exclusivement sur le choix de la métrique de classification. Il est "nonparamétrique" (seul k doit être fixé) et se base uniquement sur les données d'entraînement. L'idée est la suivante : à partir d'une base de données étiquetées, on peut estimer la classe d'une nouvelle donnée en regardant quelle est la classe majoritaire des k données voisines les plus proches (d'où le nom de l'algorithme)

L'arbre de décision :

L'arbre de décision est un algorithme appliqué sur un arbre orienté dont Les noeuds internes sont étiquetés par un test applicable à tout individu, généralement sur un attribut de description. Les arcs contiennent les résultats du test. Les feuilles sont étiquetées par une classe par défaut2 qualités principales : Facilement interprétables Classification rapide

NaiveBayes :

Le naive Bayes classifier se base sur le théorème de Bayes. Ce dernier est un classique de la théorie des probabilités. Ce théorème est fondé sur les probabilités conditionnelles.

L'avantage du classifieur bayésien naïf est qu'il requiert relativement peu de données d'entraînement pour estimer les paramètres nécessaires à la classification, à savoir moyennes et variances des différentes variables.

Régression logistique :

Le principe du modèle de la régression logistique est de relier la survenance ou la non survenance d'un événement au niveau de variables explicatives. Par exemple, dans le domaine phytosanitaire, on cherche à évaluer à partir de quelle dose d'un agent chimique, un insecte sera neutralisé.

Forêt aléatoire :

Un random forest est constitué d'un ensemble d'arbres de décision indépendants. Chaque arbre dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire : un tirage aléatoire avec remplacement sur les observations (les lignes de votre base de données). Ce processus s'appelle le tree bagging, un tirage aléatoire sur les variables (les colonnes de votre base de données). Ce processus s'appelle le feature sampling.

A la fin, tous ces arbres de décisions indépendants sont assemblés. La prédiction faite par le random forest pour des données inconnues est alors la moyenne (ou le vote, dans le cas d'un problème de classification) de tous les arbres.

XGboost :

L'algorithme XGBoost est un algorithme ensembliste qui agrège des arbres. À chaque itération, le nouvel arbre apprend de l'erreur commise par l'arbre précédent. Ainsi, même si chaque arbre a un pouvoir prédictif faible, la règle de décision construite en sommant le résultat de chaque arbre est elle très fiable

Support vector machine(SVM) :

Le modèle SVM a été construit à l'aide de la fonction svm importée de sklearn en python. Dans le modèle Support Vector Machine, le noyau la fonction est le paramètre le plus important. Il est utile lorsque le l'ensemble de données est volumineux. Le principal avantage du noyau linéaire était un traitement rapide.

- ➔ Maintenant dans cette partie on va appliquer nos modèles qu'on vient de les annoncer ci-dessus sur les variables qu'on a sélectionnées par les différentes méthodes.

1. Application des modeles en utilisant la variance des variables:

1.1 Modele KNeighbors Classifier (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
knn_model=KNeighborsClassifier(n_neighbors=36,metric='euclidean')

knn_model.fit(x_train,y_train)

print('score du train ',knn_model.score(x_train,y_train))
print('score du test ',knn_model.score(x_test,y_test))

score du train  0.8080048760666396
score du test 0.7962085308056872
```

Figure : Score de KNN avec variances entre variables

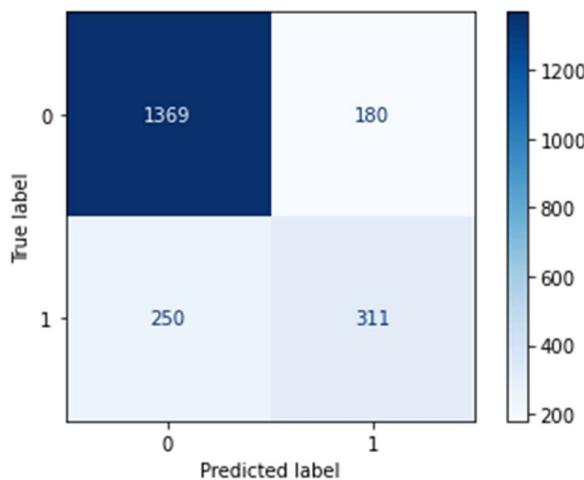


Figure : Matrice de confusion de KNN avec variances entre variables

1.1.1 Rapport de classification du KNN

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.845584 | 0.883796 | 0.864268 | 1549 |
| 1 | 0.633401 | 0.554367 | 0.591255 | 561 |
| accuracy | | | 0.796209 | 2110 |
| macro avg | 0.739492 | 0.719082 | 0.727761 | 2110 |
| weighted avg | 0.789169 | 0.796209 | 0.791680 | 2110 |

Figure : Rapport de classification de KNN avec variances entre variables

1.1.2 Courbe Roc du KNN

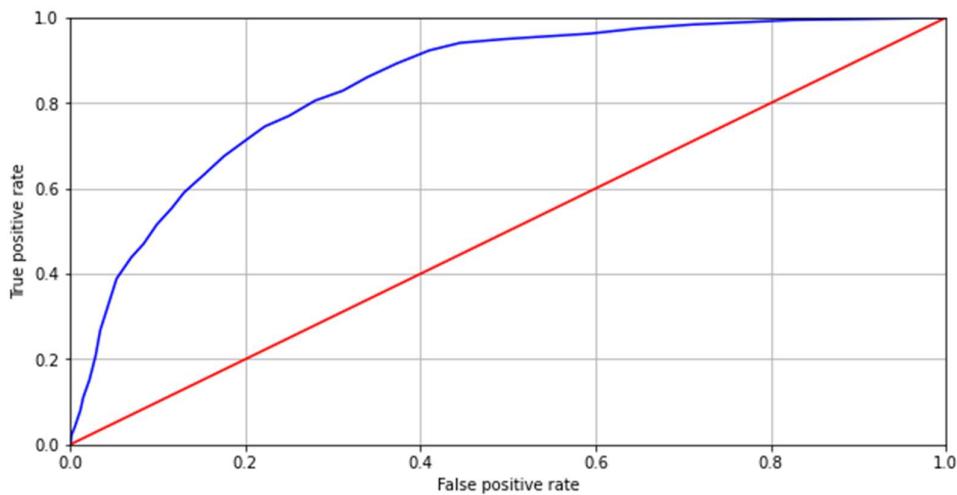


Figure : Courbe ROC de KNN avec variances entre variables

1.2 Modele DecisionTreeClassifier(DT)

```
from sklearn.tree import DecisionTreeClassifier  
  
tree_model=DecisionTreeClassifier(criterion='gini',max_depth=7,random_state=0)  
tree_model.fit(x_train,y_train)  
print('train score : ', tree_model.score(x_train,y_train))  
print('test score : ', tree_model.score(x_test,y_test))  
  
train score :  0.8228362454286875  
test score :  0.79478672985782
```

Figure : Score de DT avec variances entre variables

1.2.1 Matrice de confusion de DT

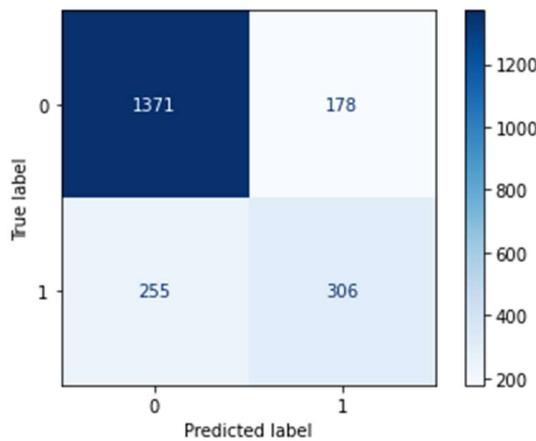


Figure : Matrice de confusion de DT avec variances entre variables

1.2.2 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.843173 | 0.885087 | 0.863622 | 1549 |
| 1 | 0.632231 | 0.545455 | 0.585646 | 561 |
| accuracy | | | 0.794787 | 2110 |
| macro avg | 0.737702 | 0.715271 | 0.724634 | 2110 |
| weighted avg | 0.787089 | 0.794787 | 0.789715 | 2110 |

Figure : Rapport de classification de DT avec variances entre variables

1.2.3 Courbe Roc du DT

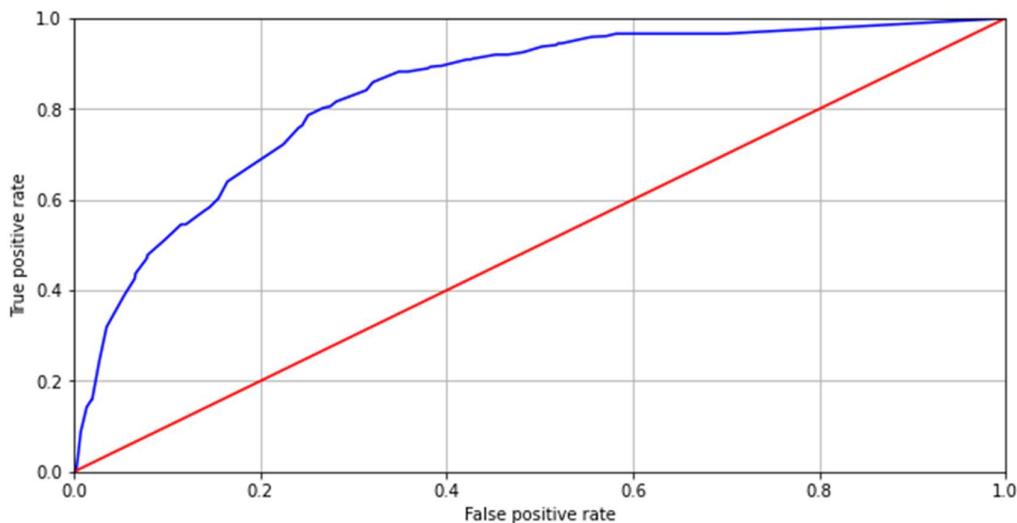


Figure : Courbe ROC de DT avec variances entre variables

1.2.4 Arbre de Decision du DT

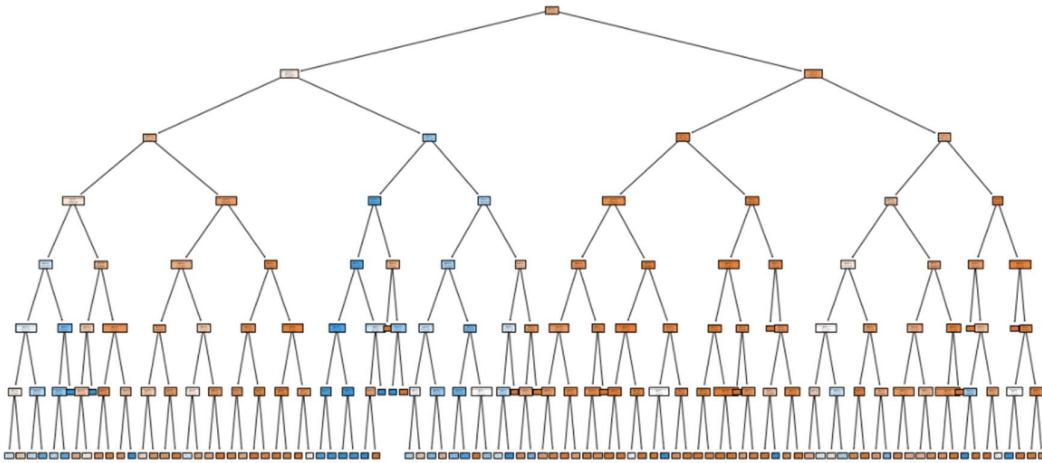


Figure : Arbre de décision avec variances entre variables

1.2.5 Text d'export du DT

```

--- Tenure Months <= 0.22
    --- Internet Service_Fiber optic <= 0.50
        |--- Tenure Months <= 0.04
            |--- Device Protection_No internet service <= 0.50
                |--- Senior Citizen_Yes <= 0.50
                    |--- Payment Method_Electronic check <= 0.50
                        |--- Tenure Months <= 0.01
                            |--- class: 1
                        |--- Tenure Months > 0.01
                            |--- class: 0
                    |--- Payment Method_Electronic check > 0.50
                        |--- Streaming Movies_Yes <= 0.50
                            |--- class: 1
                        |--- Streaming Movies_Yes > 0.50
                            |--- class: 1
                |--- Senior Citizen_Yes > 0.50
                    |--- Online Security_Yes <= 0.50
                        |--- Paperless Billing_Yes <= 0.50
                            |--- class: 1
                        |--- Paperless Billing_Yes > 0.50
                            |--- class: 1
                    |--- Online Security_Yes > 0.50
                        |--- class: 1
            |--- Device Protection_No internet service > 0.50
                |--- Tenure Months <= 0.01
                    |--- Senior Citizen_Yes <= 0.50
                        |--- Paperless Billing_Yes <= 0.50
                            |--- class: 0
                        |--- Paperless Billing_Yes > 0.50
                            |--- class: 0
                    |--- Senior Citizen_Yes > 0.50
                        |--- class: 1
                |--- Tenure Months > 0.01
                    |--- Payment Method_Credit card (automatic) <= 0.50
                        |--- Gender_Male <= 0.50
                            |--- class: 0
                        |--- Gender_Male > 0.50
                            |--- class: 1
                    |--- Payment Method_Credit card (automatic) > 0.50
                        |--- class: 1
    |--- Tenure Months > 0.04
        |--- Streaming TV_No internet service <= 0.50
            |--- Payment Method_Electronic check <= 0.50
                |--- Tenure Months <= 0.06
                    |--- Online Security_Yes <= 0.50
                        |--- class: 0
                    |--- Online Security_Yes > 0.50
                        |--- class: 0
                |--- Tenure Months > 0.06
                    |--- Dependents_Yes <= 0.50
                        |--- class: 0
                    |--- Dependents_Yes > 0.50
                        |--- class: 0
            |--- Payment Method_Electronic check > 0.50
                |--- Online Backup_Yes <= 0.50
                    |--- Online Security_Yes <= 0.50
                        |--- class: 1
                    |--- Online Security_Yes > 0.50
                        |--- class: 0
                |--- Online Backup_Yes > 0.50
                    |--- Tech Support_Yes <= 0.50
                        |--- class: 0
                    |--- Tech Support_Yes > 0.50
                        |--- class: 0
            |--- Streaming TV_No internet service > 0.50
                |--- Dependents_Yes <= 0.50
                    |--- Contract_One year <= 0.50
                        |--- Partner_Yes <= 0.50
                            |--- class: 0
                        |--- Partner_Yes > 0.50
                            |--- class: 0
                    |--- Contract_One year > 0.50
                        |--- Partner_Yes <= 0.50
                            |--- class: 0
                        |--- Partner_Yes > 0.50
                            |--- class: 0
                |--- Dependents_Yes > 0.50
                    |--- class: 1

```

Figure : Text export de DT avec variances entre variables

1.3 Modele NaiveBayes (NB)

```

: from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
: from sklearn.model_selection import cross_val_score
: from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
      'bernoulli': BernoulliNB(),
      'multinomial': MultinomialNB()}

scores = []
for key, model in nb.items(): #nb.items; parcourir cle et valeur
    s = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)

scores
: {'gaussian': 0.694220420122983,
 'bernoulli': 0.7352612356072801,
 'multinomial': 0.7421707729767653}

: bayes_model = MultinomialNB()
bayes_model.fit(x_train, y_train)
print('train score : ', bayes_model.score(x_train,y_train))
print('test score : ', bayes_model.score(x_test,y_test))

train score :  0.7423811458756603
test score :  0.7568720379146919

```

Figure : Score de NB avec variances entre variables

1.3.1 Matrice de confusion du NB

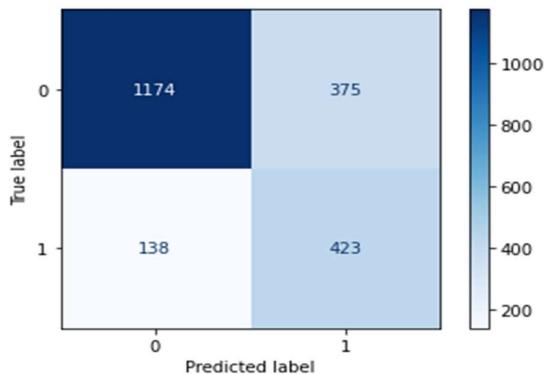


Figure : Matrice de confusion de NB avec variances entre variables

1.3.2 Rapport de classification du NB

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.89482 | 0.75791 | 0.82069 | 1549 |
| 1 | 0.53008 | 0.75401 | 0.62252 | 561 |
| accuracy | | | 0.75687 | 2110 |
| macro avg | 0.71245 | 0.75596 | 0.72160 | 2110 |
| weighted avg | 0.79784 | 0.75687 | 0.76800 | 2110 |

Figure : Rapport de classification de NB avec variances entre variables

1.3.3 Courbe Roc du NB

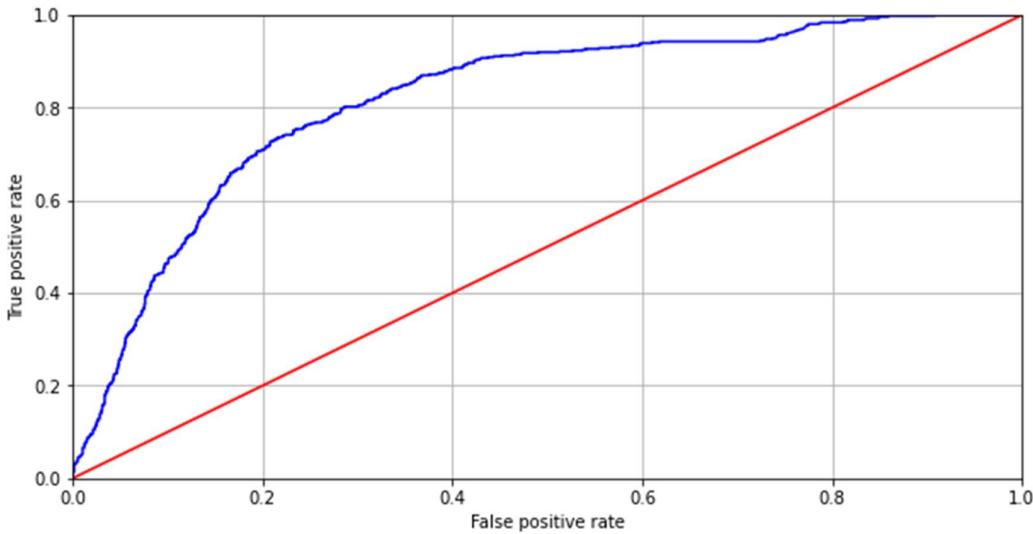


Figure : Courbe ROC de NB avec variances entre variables

1.4 Modele LogisticRegression (LR)

```
logre_model = LogisticRegression(random_state=0, C=101., penalty = 'l2', solver='lbfgs')
logre_model.fit(x_train, y_train)

print('train score : ', logre_model.score(x_train, y_train) )
print('test score : ', logre_model.score(x_test,y_test) )

train score :  0.8096302316131654
test score :  0.8104265402843602
```

Figure : Score de LR avec variances entre variables

1.4.1 Matrice de confusion du LR

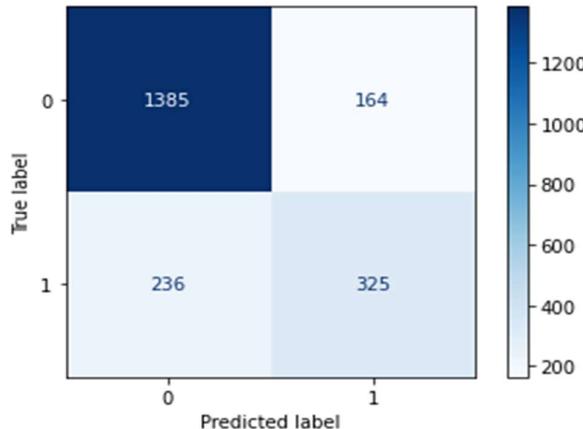


Figure : Matrice de confusion de LR avec variances entre variables

1.4.2 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85441 | 0.89413 | 0.87382 | 1549 |
| 1 | 0.66462 | 0.57932 | 0.61905 | 561 |
| accuracy | | | 0.81043 | 2110 |
| macro avg | 0.75952 | 0.73672 | 0.74643 | 2110 |
| weighted avg | 0.80395 | 0.81043 | 0.80608 | 2110 |

Figure : Rapport de classification de LR avec variances entre variables

1.4.3 Courbe Roc du LR

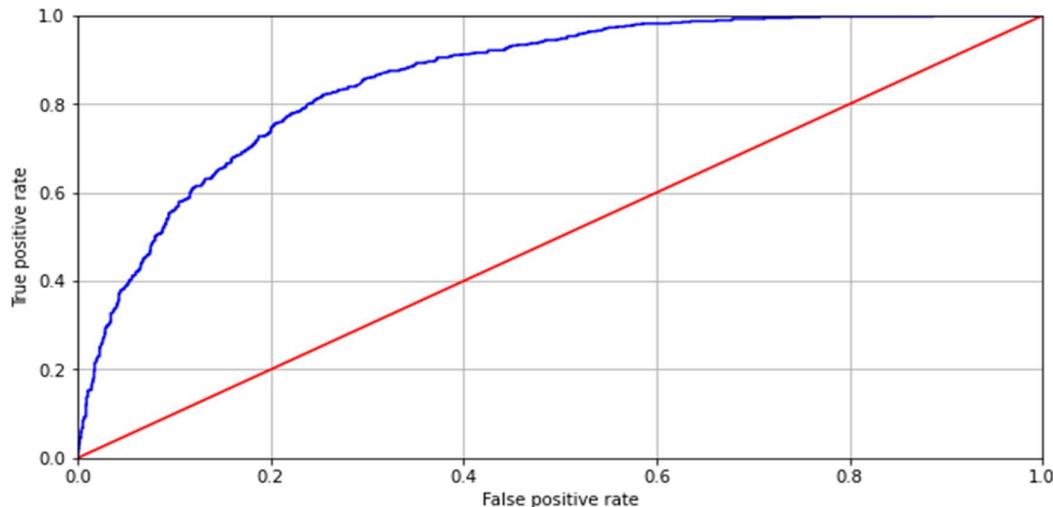


Figure : Courbe ROC de LR avec variances entre variables

1.5 Modele RandomForestClassifier (RFC)

```
from sklearn.ensemble import RandomForestClassifier  
  
random_model=RandomForestClassifier(criterion='gini', max_depth=8,random_state=14)  
random_model.fit(x_train,y_train)  
  
print('train score : ',random_model.score(x_train,y_train))  
print('test score : ',random_model.score(x_test,y_test))
```

```
train score :  0.8283218203982121  
test score :  0.8099526066350711
```

Figure : Score de RFC avec variances entre variables

1.5.1 Matrice de confusion du RFC

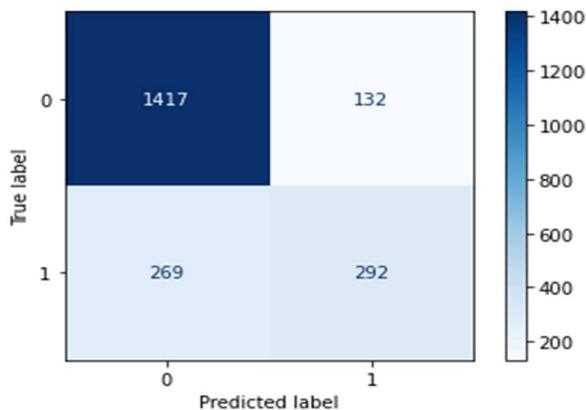


Figure : Matrice de confusion de RFC avec variances entre variables

1.5.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84045 | 0.91478 | 0.87604 | 1549 |
| 1 | 0.68868 | 0.52050 | 0.59289 | 561 |
| accuracy | | | 0.80995 | 2110 |
| macro avg | 0.76457 | 0.71764 | 0.73447 | 2110 |
| weighted avg | 0.80010 | 0.80995 | 0.80076 | 2110 |

Figure : Rapport de classification de RFC avec variances entre variables

1.5.3 Courbe Roc du RFC

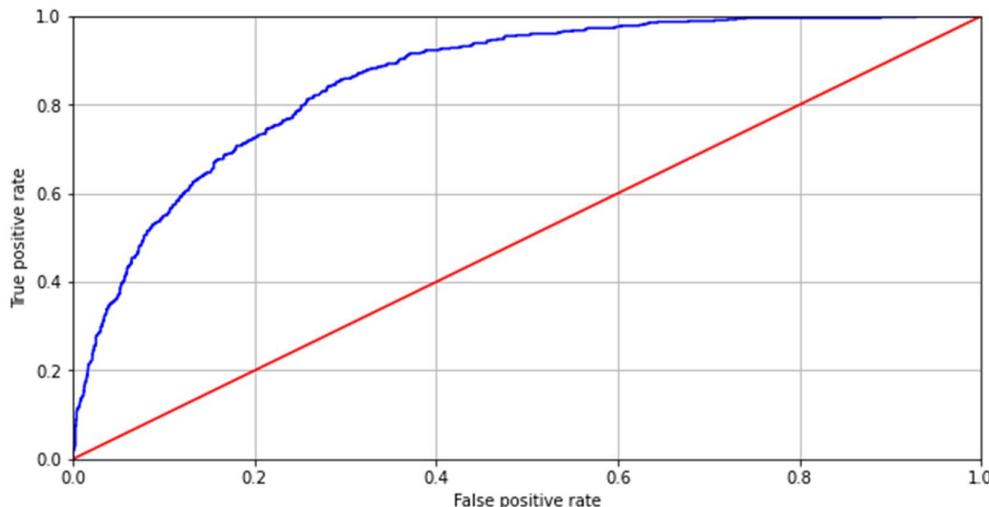


Figure : Courbe ROC de RFC avec variances entre variables

1.6 Modele XGboost(xgb)

```
from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=4,n_estimators=60,learning_rate=0.1)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

C:\Users\User\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use
l be removed in a future release. To remove this warning, do the following: 1) Pass opt
fier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, .
warnings.warn(label_encoder_deprecation_msg, UserWarning)
[13:02:22] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/le
valuation metric used with the objective 'binary:logistic' was changed from 'error' to
restore the old behavior.
0.816
0.8123667377398721
```

Figure : Score de XGb avec variances entre variables

1.6.1 Matrice de confusion du xgb

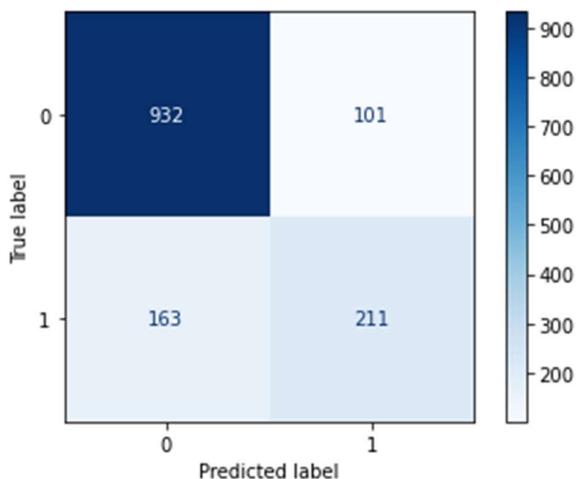


Figure : Matrice de confusion deXGB avec variances entre variables

1.6.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85114 | 0.90223 | 0.87594 | 1033 |
| 1 | 0.67628 | 0.56417 | 0.61516 | 374 |
| accuracy | | | 0.81237 | 1407 |
| macro avg | 0.76371 | 0.73320 | 0.74555 | 1407 |
| weighted avg | 0.80466 | 0.81237 | 0.80662 | 1407 |

Figure : Rapport de classification de XGB avec variances entre variables

1.6.3 Courbe Roc du xgb

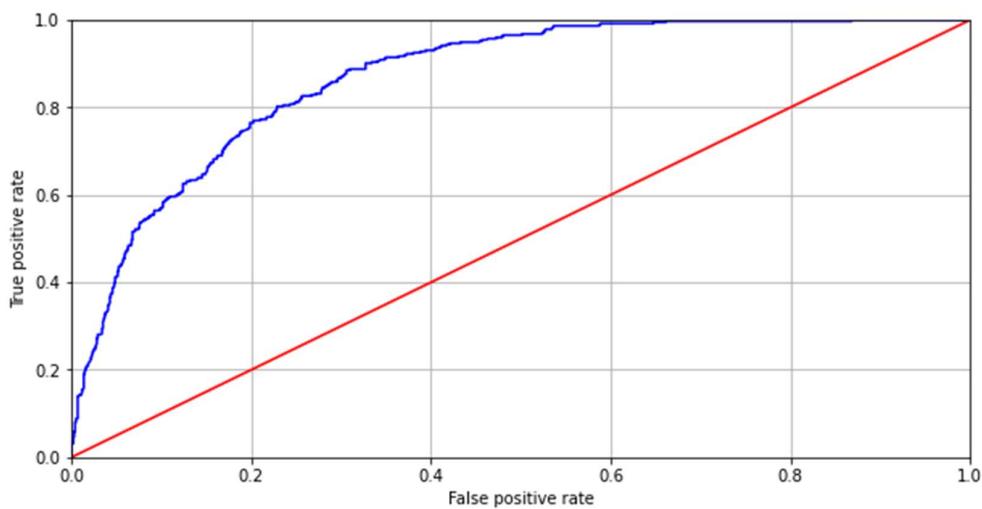


Figure : Courbe ROC de XGB avec variances entre variables

1.7 Modele Support Vector Machine (SVM)

```

from sklearn import svm

svm_model=svm.SVC(C=10,kernel='rbf',gamma=0.01)
svm_model.fit(x_train,y_train)
print(svm_model.score(x_train,y_train))
print(svm_model.score(x_test,y_test))

0.8074666666666667
0.8180525941719972

```

Figure : Score de SVM avec variances entre variables

1.7.1 Matrice de confusion du SVM

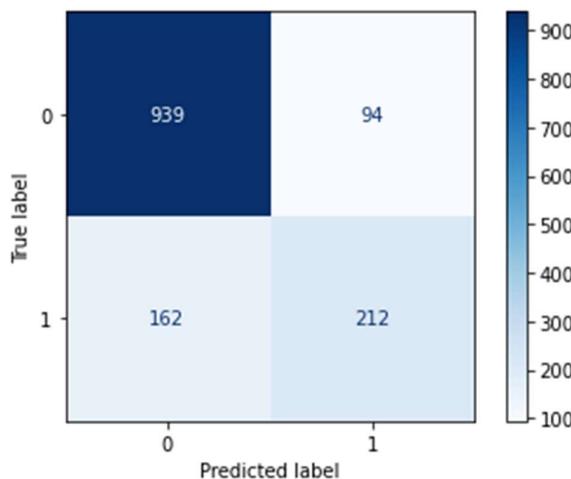


Figure : Matrice de confusion de SVM avec variances entre variables

1.7.2 Rapport de classification du SVM

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85286 | 0.90900 | 0.88004 | 1033 |
| 1 | 0.69281 | 0.56684 | 0.62353 | 374 |
| accuracy | | | 0.81805 | 1407 |
| macro avg | 0.77284 | 0.73792 | 0.75178 | 1407 |
| weighted avg | 0.81032 | 0.81805 | 0.81185 | 1407 |

Figure : Rapport de classification de SVM avec variances entre variables

1.8 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8080 | 0.7962 | 0.8642 | 0.5912 | 0.7962 | 0.8433 |
| DecisionTreeClassifier | 0.8228 | 0.7947 | 0.8636 | 0.5856 | 0.7947 | 0.8366 |
| NaiveBayes(MultiNomial) | 0.7423 | 0.7568 | 0.8206 | 0.6225 | 0.7568 | 0.8192 |
| LogisticRegression | 0.8096 | 0.8104 | 0.8738 | 0.6190 | 0.8104 | 0.8586 |
| RandomForestClassifier | 0.8283 | 0.8099 | 0.8760 | 0.5928 | 0.8099 | 0.8579 |
| XGBClassifier | 0.8116 | 0.8123 | 0.8759 | 0.6151 | 0.8123 | 0.8685 |
| SVM | 0.8074 | 0.8081 | 0.8800 | 0.6235 | 0.8180 | False |

Variance (with 26 features)

Figure : Table de résumé avec variances entre variables

- Le modèle XGBClassifier donne les meilleurs résultats si on utilise la variance pour la sélection des variables (avec 26 variables restantes)

2 Application des modeles en utilisant SelectKBest

2.1 Modele KNeighbors Classifier (KNN)

```
: from sklearn.neighbors import KNeighborsClassifier

knn_model=KNeighborsClassifier(n_neighbors=47,metric='euclidean')
knn_model.fit(x_train,y_train)

print('score du train ' ,knn_model.score(x_train,y_train))
print('score du test' ,knn_model.score(x_test,y_test))

score du train  0.8035351483136937
score du test 0.8014218009478673
```

Figure : Score de KNN avec SelectKBest

2.1.2 Matrice de confusion du KNN

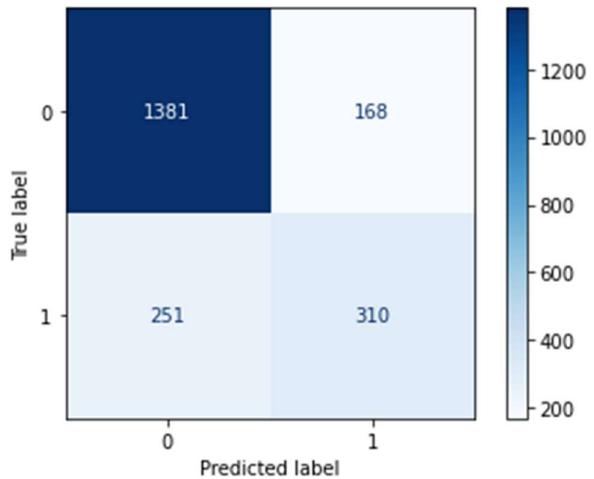


Figure : Matrice de confusion de KNN avec SelektKBest

2.1.3 Rapport de classification du KNN

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.846201 | 0.891543 | 0.868280 | 1549 |
| 1 | 0.648536 | 0.552585 | 0.596728 | 561 |
| accuracy | | | 0.801422 | 2110 |
| macro avg | 0.747368 | 0.722064 | 0.732504 | 2110 |
| weighted avg | 0.793646 | 0.801422 | 0.796081 | 2110 |

Figure : Rapport de classification de KNN avec SelektKBest

2.1.4 Courbe Roc du KNN

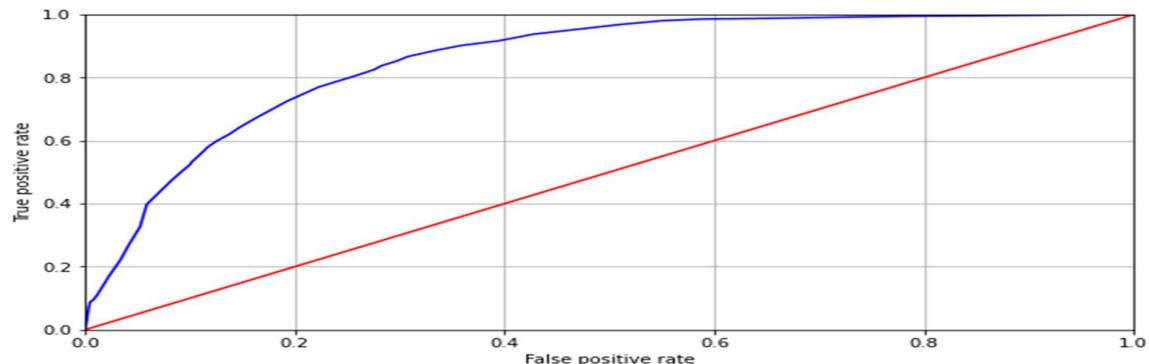


Figure Courbe ROC de KNN avec SelektKBest

2.2 Modele DecisionTreeClassifier(DT)

```

: from sklearn.tree import DecisionTreeClassifier

tree_model=DecisionTreeClassifier(criterion='gini',max_depth=6,random_state=0)
tree_model.fit(x_train,y_train)
print('train score : ', tree_model.score(x_train,y_train))
print('test score : ', tree_model.score(x_test,y_test))

train score :  0.8071921982933766
test score :  0.7962085308056872

```

Figure : Score de DT avec SelectKBest

2.2.1 Matrice de confusion de DT

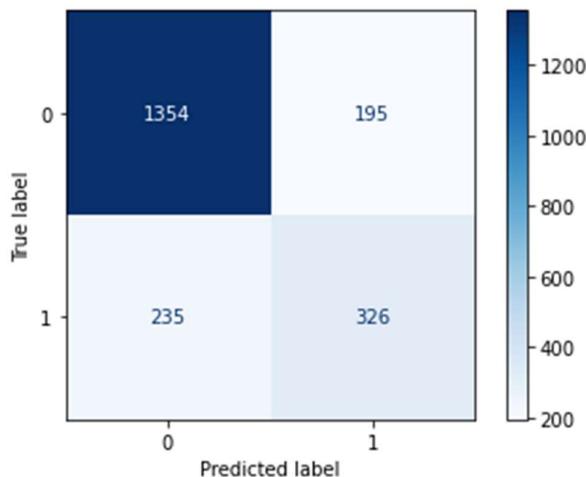


Figure : Matrice de confusion de DT avec SelektKBest

2.2.2 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|----------|
| 0 | 0.852108 | 0.874112 | 0.862970 | 1549 |
| 1 | 0.625720 | 0.581105 | 0.602588 | 561 |
| accuracy | | | | 0.796209 |
| macro avg | 0.738914 | 0.727609 | 0.732779 | 2110 |
| weighted avg | 0.791917 | 0.796209 | 0.793740 | 2110 |

Figure : Rapport de classification de DT avec SelektKBest

2.2.3 Courbe Roc du DT

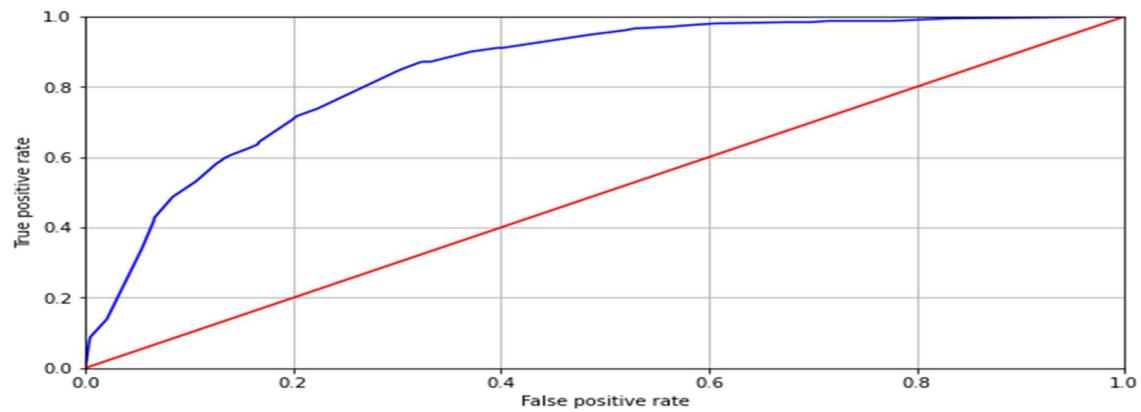


Figure : Courbe ROC de DT avec SelektKBest

2.2.4 Arbre de Decision du DT

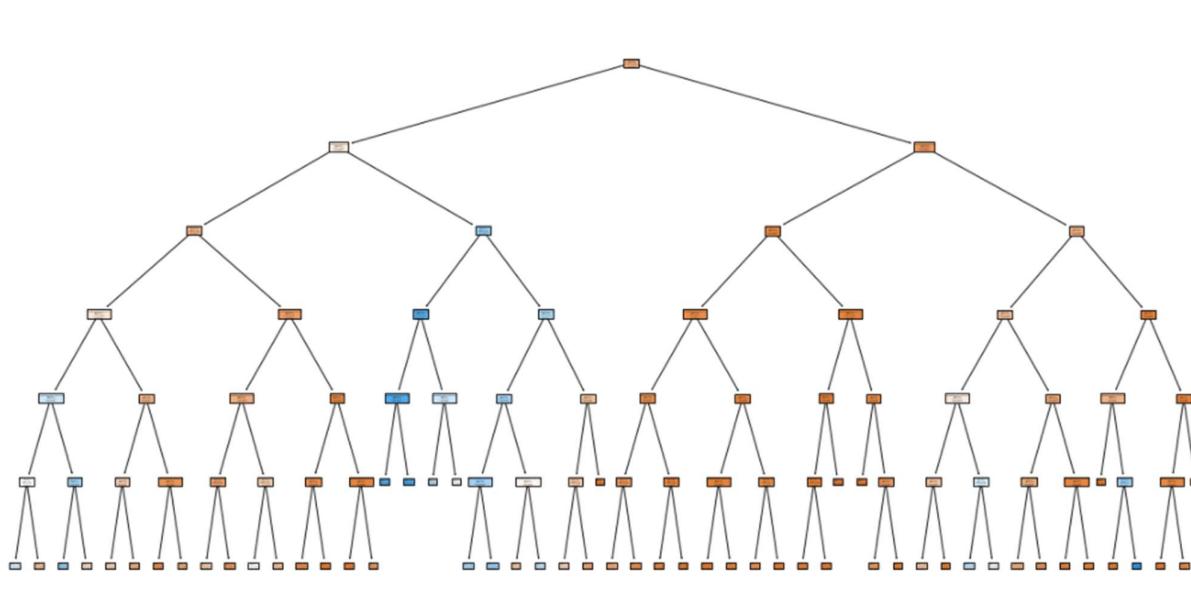


Figure : Arbre de décision avec SelektKBest

2.2.5 Text d'export du DT

```

|--- Tenure Months <= 0.22
|--- Internet Service_Fiber optic <= 0.50
|   |--- Tenure Months <= 0.04
|   |   |--- Streaming TV_No internet service <= 0.50
|   |   |   |--- Payment Method_Electronic check <= 0.50
|   |   |   |   |--- Tenure Months <= 0.02
|   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Tenure Months > 0.02
|   |   |   |   |   |--- class: 0
|   |   |   |--- Payment Method_Electronic check > 0.50
|   |   |   |   |--- Dependents_Yes <= 0.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Dependents_Yes > 0.50
|   |   |   |   |   |--- class: 0
|--- Streaming TV_No internet service > 0.50
|   |--- Tenure Months <= 0.01
|   |   |--- Dependents_Yes <= 0.50
|   |   |   |--- class: 0
|   |   |   |--- Dependents_Yes > 0.50
|   |   |   |--- class: 0
|--- Tenure Months > 0.01
|   |--- Payment Method_Electronic check <= 0.50
|   |   |--- class: 0
|   |--- Payment Method_Electronic check > 0.50
|   |   |--- class: 0
|--- Tenure Months > 0.04
|--- Tech Support_No internet service <= 0.50
|   |--- Payment Method_Electronic check <= 0.50
|   |   |--- Tenure Months <= 0.06
|   |   |   |--- class: 0
|   |   |   |--- Tenure Months > 0.06
|   |   |   |--- class: 0
|   |--- Payment Method_Electronic check > 0.50
|   |   |--- Tenure Months <= 0.06
|   |   |   |--- class: 0
|--- Dependents_Yes > 0.50
|   |--- Contract_Two year <= 0.50
|   |   |--- class: 0
|   |--- Contract_Two year > 0.50
|   |   |--- class: 0
|--- Dependents_Yes > 0.50
|   |--- Payment Method_Electronic check <= 0.50
|   |   |--- class: 0
|   |--- Payment Method_Electronic check > 0.50
|   |   |--- class: 0
|--- Internet Service_Fiber optic > 0.50
|--- Tenure Months < 0.01
|   |--- Dependents_Yes <= 0.50
|   |   |--- Payment Method_Electronic check <= 0.50
|   |   |   |--- class: 1
|   |   |   |--- Payment Method_Electronic check > 0.50
|   |   |   |--- class: 1
|   |--- Dependents_Yes > 0.50
|   |   |--- Payment Method_Electronic check <= 0.50
|   |   |   |--- class: 1
|   |   |   |--- Payment Method_Electronic check > 0.50
|   |   |   |--- class: 0
|--- Tenure Months > 0.01
|   |--- Dependents_Yes <= 0.50
|   |   |--- Tenure Months <= 0.20
|   |   |   |--- Payment Method_Electronic check <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |   |--- Payment Method_Electronic check > 0.50
|   |   |   |   |--- class: 1
|   |--- Tenure Months > 0.20
|   |   |--- Payment Method_Electronic check <= 0.50
|   |   |   |--- class: 0
|   |   |   |--- Payment Method_Electronic check > 0.50
|   |   |   |--- class: 1
|--- Dependents_Yes > 0.50
|   |--- Tenure Months <= 0.20

```

Figure : Text export de DT avec SelektKBest

2.3 Modele NaiveBayes (NB)

```

!5]: from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
       'bernoulli': BernoulliNB(),
       'multinomial': MultinomialNB()}
scores = {}
for key, model in nb.items():  #nb.items; parcourir cle et valeur
    s = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)
scores

!5]: {'gaussian': 0.6671984235070777,
       'bernoulli': 0.7348570013618917,
       'multinomial': 0.7777283232223186}

!6]: bayes_model = MultinomialNB()
bayes_model.fit(x_train, y_train)
print('train score : ', bayes_model.score(x_train,y_train))
print('test score : ', bayes_model.score(x_test,y_test))

train score :  0.778138967899228
test score :  0.7758293838862559

```

Figure : Score de NB avec SelectKBest

2.3.1 Matrice de confusion du NB

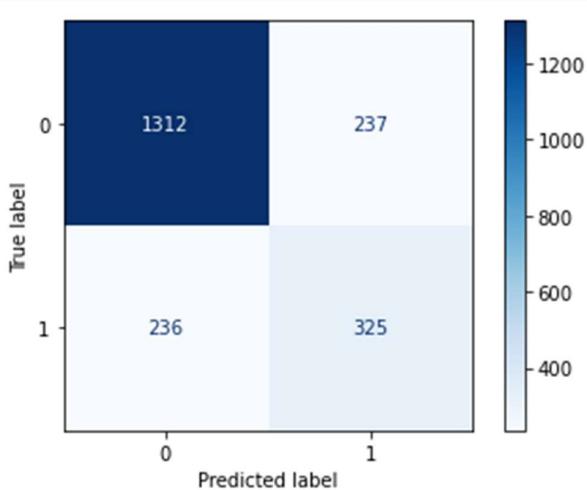


Figure : Matrice de confusion de NB avec SelektKBest

2.3.2 Rapport de classification du NB

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84755 | 0.84700 | 0.84727 | 1549 |
| 1 | 0.57829 | 0.57932 | 0.57881 | 561 |
| accuracy | | | 0.77583 | 2110 |
| macro avg | 0.71292 | 0.71316 | 0.71304 | 2110 |
| weighted avg | 0.77596 | 0.77583 | 0.77589 | 2110 |

Figure : Rapport de classification de NB avec SelektKBest

2.3.4 Courbe Roc du NB

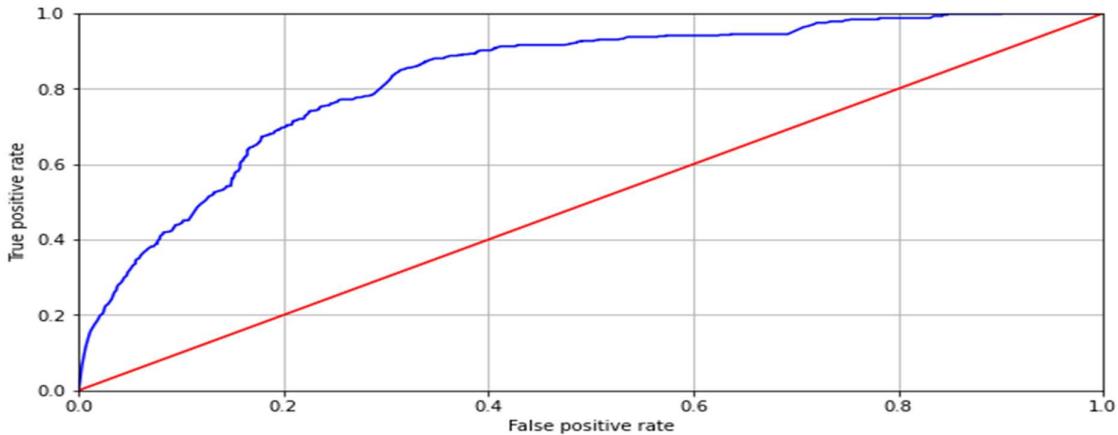


Figure : Courbe ROC de NB avec SelektKBest

2.4 Modele LogisticRegression (LR)

```
: from sklearn.linear_model import LogisticRegression  
  
logre_model = LogisticRegression(random_state=0, C=0.10101, penalty = 'l1', solver='saga')  
logre_model.fit(x_train, y_train)  
  
print('train score : ' , logre_model.score(x_train, y_train) )  
print('test score : ' , logre_model.score(x_test,y_test) )  
  
train score :  0.8015034538805363  
test score :  0.8052132701421801
```

Figure : Score de LR avec SelectKBest

2.4.1 Matrice de confusion du LR

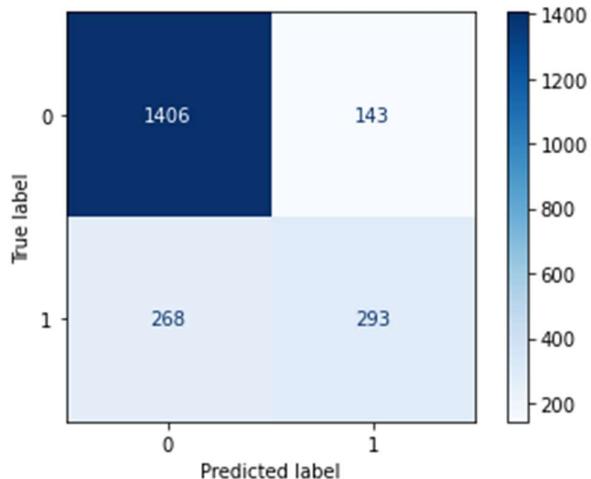


Figure : Matrice de confusion de LR avec SelektKBest

2.4.2 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83990 | 0.90768 | 0.87248 | 1549 |
| 1 | 0.67202 | 0.52228 | 0.58776 | 561 |
| accuracy | | | 0.80521 | 2110 |
| macro avg | 0.75596 | 0.71498 | 0.73012 | 2110 |
| weighted avg | 0.79527 | 0.80521 | 0.79678 | 2110 |

Figure : Rapport de classification de LR avec SelektKBest

2.4.3 Courbe Roc du LR

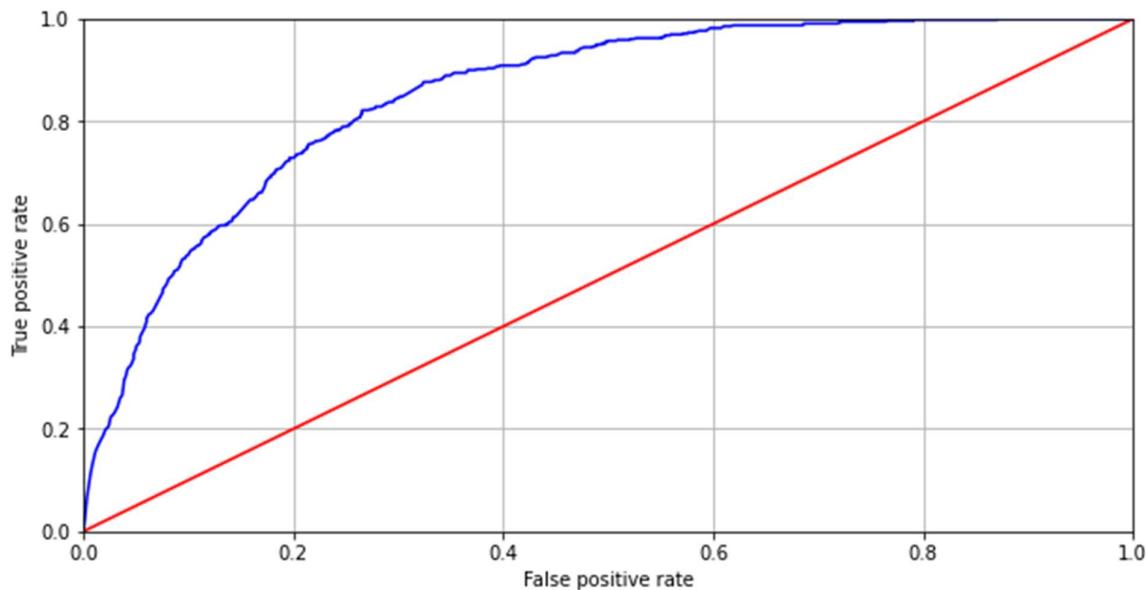


Figure Courbe ROC de LR avec SelektKBest

2.5 Modele RandomForestClassifier (RFC)

```
: from sklearn.ensemble import RandomForestClassifier  
  
random_model=RandomForestClassifier(criterion='gini', max_depth=9,random_state=32)  
random_model.fit(x_train,y_train)  
  
print('train score : ',random_model.score(x_train,y_train))  
print('test score : ',random_model.score(x_test,y_test))  
  
train score :  0.8152888888888888  
test score :  0.7853589196872779
```

Figure : Score de RFC avec SelectKBest

2.5.1 Matrice de confusion du RFC

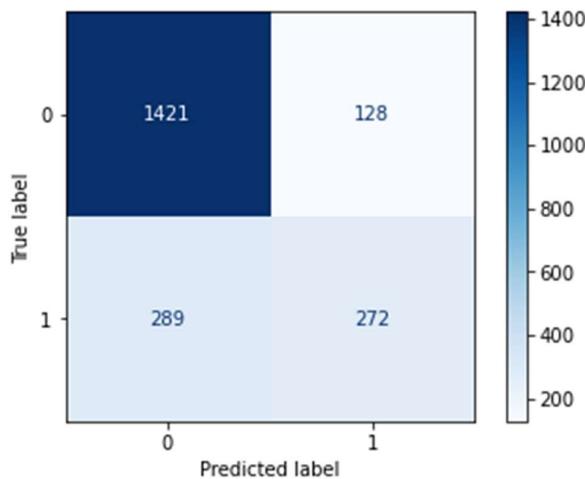


Figure : Matrice de confusion de RFC avec SelektKBest

2.5.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.82984 | 0.91930 | 0.87228 | 1549 |
| 1 | 0.68274 | 0.47950 | 0.56335 | 561 |
| accuracy | | | 0.80237 | 2110 |
| macro avg | 0.75629 | 0.69940 | 0.71782 | 2110 |
| weighted avg | 0.79073 | 0.80237 | 0.79014 | 2110 |

Figure : Rapport de classification de RFC avec SelektKBest

2.5.3 Courbe Roc du RFC

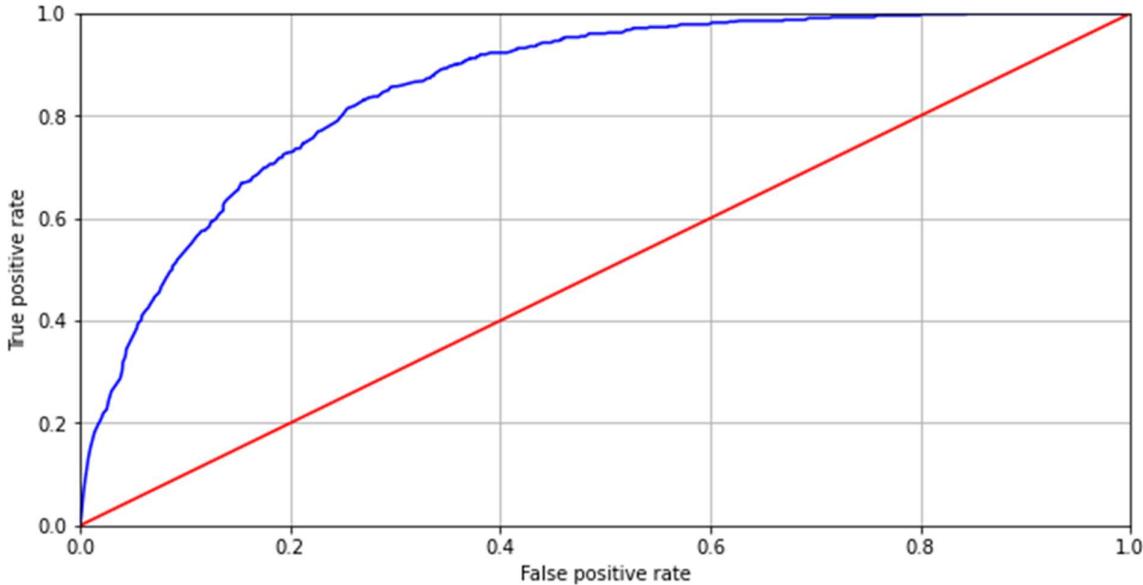


Figure : Courbe ROC de RFC avec SelektKBest

2.6 Modele XGboost(xgb)

```

from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=2,n_estimators=180,learning_rate=0.05)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

```

[23:08:27] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.
valuation metric used with the objective 'binary:logistic' was changed from 'error' to 'loglc
restore the old behavior.
0.8012444444444444
0.8024164889836531

Figure : Score de XGB avec SelectKBest

2.6.1 Matrice de confusion du xgb

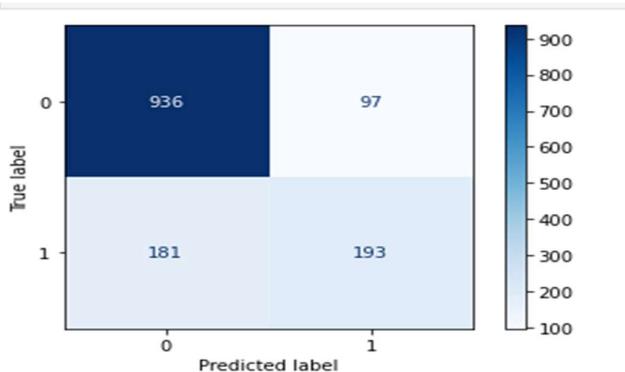


Figure : Matrice de confusion deXGB avec SelektKBest

2.6.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83796 | 0.90610 | 0.87070 | 1033 |
| 1 | 0.66552 | 0.51604 | 0.58133 | 374 |
| accuracy | | | 0.80242 | 1407 |
| macro avg | 0.75174 | 0.71107 | 0.72601 | 1407 |
| weighted avg | 0.79212 | 0.80242 | 0.79378 | 1407 |

Figure : Rapport de classification de XGB avec SelektKBest

2.6.3 Courbe Roc du xgb

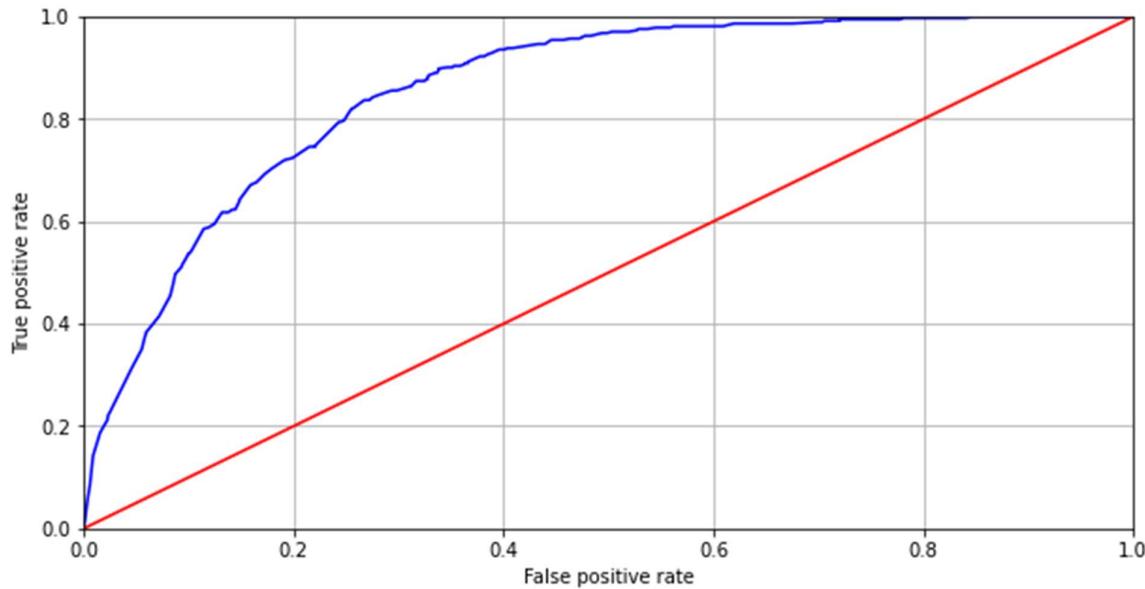


Figure : Courbe ROC de XGB avec SelektKBest

2.7 Modele Support Vector Machine (SVM)

```
from sklearn import svm

svm_model=svm.SVC(C=10,kernel='rbf',gamma=0.001)
svm_model.fit(x_train,y_train)
print(svm_model.score(x_train,y_train))
print(svm_model.score(x_test,y_test))

0.8019555555555555
0.8052594171997157
```

Figure : Score de SVM avec SelectKBest

2.7.1 Matrice de confusion du SVM

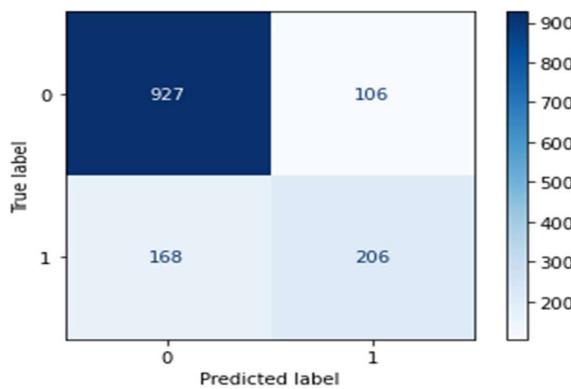


Figure : Matrice de confusion de SVM avec SelektKBest

2.7.2 Rapport de classification du SVM

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84658 | 0.89739 | 0.87124 | 1033 |
| 1 | 0.66026 | 0.55080 | 0.60058 | 374 |
| accuracy | | | 0.80526 | 1407 |
| macro avg | 0.75342 | 0.72409 | 0.73591 | 1407 |
| weighted avg | 0.79705 | 0.80526 | 0.79930 | 1407 |

Figure : Rapport de classification de SVM avec SelektKBest

2.8 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8035 | 0.8014 | 0.8682 | 0.5967 | 0.8014 | 0.8545 |
| DecisionTreeClassifier | 0.8071 | 0.7962 | 0.8629 | 0.6025 | 0.7962 | 0.8448 |
| NaiveBayes(MultiNomial) | 0.7781 | 0.7758 | 0.8472 | 0.5788 | 0.7758 | 0.8272 |
| LogisticRegression | 0.8015 | 0.8052 | 0.8724 | 0.5877 | 0.8052 | 0.8524 |
| RandomForestClassifier | 0.8029 | 0.8023 | 0.8722 | 0.5633 | 0.8023 | 0.8574 |
| XGBClassifier | 0.8012 | 0.8024 | 0.8707 | 0.5813 | 0.8024 | 0.8587 |
| SVM | 0.8019 | 0.8052 | 0.8712 | 0.6005 | 0.8052 | False |

SelectKBest (with 8 features)

Figure : Résumé des modèles avec SelectKBest

→ Le modèle SVM donne les meilleurs résultats si on utilise SelectKBest pour la sélection des variables (avec 8 variables restantes)

3 Application des modeles en utilisant Recursive Feature Elimination (RFE)

3.1 Modele DecisionTreeClassifier(DT)

```

: from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import RFE
model = DecisionTreeClassifier()
rfe = RFE(model, n_features_to_select=9)
rfe.fit(x_train, y_train)
print("Sélection de variables", rfe.support_)
print("Classement de variables", rfe.ranking_)
print("Variables sélectionnées : ", list(x.columns[rfe.support_]))

Sélection de variables [ True  True  True  True False False False  True False False
False False False False False False False False False False False False False
False False False True True True]
Classement de variables [ 1  1  1  1 16 18  6  1 17 19  4 22  2  8 13 20  5 15 12 21 10 11 14  1
 7  3  9  1  1  1]
Variables sélectionnées : ['Gender_Male', 'Senior Citizen_Yes', 'Partner_Yes', 'Dependents_Yes', 'Internet Service_Fiber optic', 'Paperless Billing_Yes', 'Tenure Months', 'Monthly Charges', 'Total Charges']

: x_train = rfe.transform(x_train)
x_test = rfe.transform(x_test)
print(x_train.shape)
print(x_test.shape)

```

```

tree_model=DecisionTreeClassifier(criterion='entropy',max_depth=5,random_state=0)
tree_model.fit(x_train,y_train)
print('train score : ', tree_model.score(x_train,y_train))
print('test score : ', tree_model.score(x_test,y_test))

```

train score : 0.8051605038602194
test score : 0.7995260663507109

Figure : Score de DT avec RFE

3.1.1 Matrice de confusion de DT

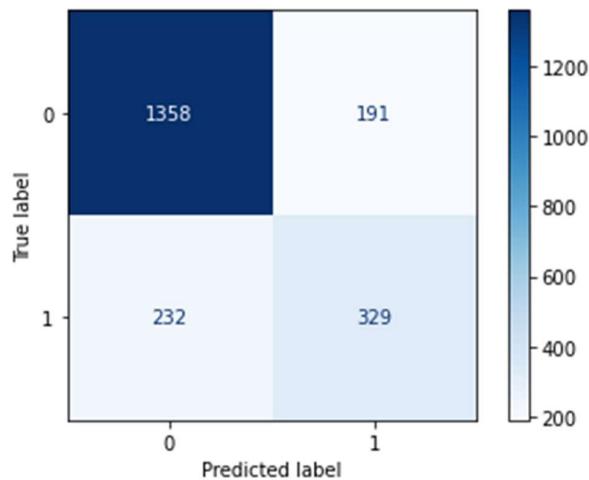


Figure : Matrice de confusion de DT avec RFE

3.1.2 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.854088 | 0.876695 | 0.865244 | 1549 |
| 1 | 0.632692 | 0.586453 | 0.608696 | 561 |
| accuracy | | | 0.799526 | 2110 |
| macro avg | 0.743390 | 0.731574 | 0.736970 | 2110 |
| weighted avg | 0.795224 | 0.799526 | 0.797034 | 2110 |

Figure : Rapport de classification de DT avec RFE

3.1.3 Courbe Roc du DT

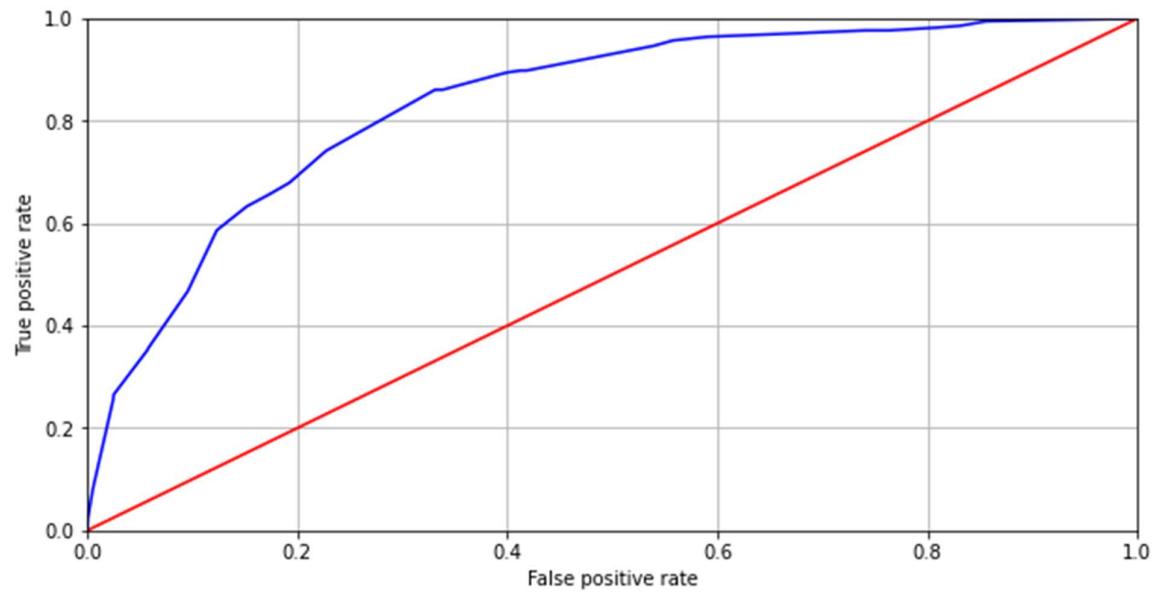


Figure : Courbe ROC de DT avec RFE

3.1.4 Arbre de Decision du DT

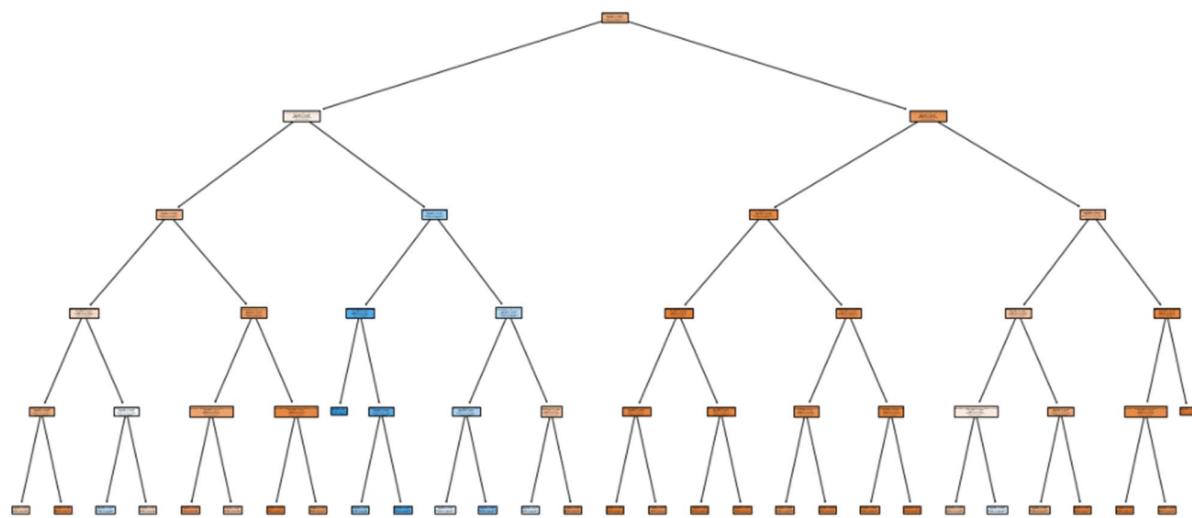


Figure : Arbre de décision avec RFE

3.1.5 Text d'export du DT

```

--- Tenure Months <= 0.22
    --- Internet Service_Fiber optic <= 0.50
        --- Tenure Months <= 0.06
            --- Monthly Charges <= 0.06
                --- Total Charges <= 0.00
                    --- class: 0
                --- Total Charges > 0.00
                    --- class: 0
            --- Monthly Charges > 0.06
                --- Total Charges <= 0.01
                    --- class: 1
                --- Total Charges > 0.01
                    --- class: 0
        --- Tenure Months > 0.06
            --- Dependents_Yes <= 0.50
                --- Payment Method_Electronic check <= 0.50
                    --- class: 0
                --- Payment Method_Electronic check > 0.50
                    --- class: 0
            --- Dependents_Yes > 0.50
                --- Payment Method_Electronic check <= 0.50
                    --- class: 0
                --- Payment Method_Electronic check > 0.50
                    --- class: 0
    --- Internet Service_Fiber optic > 0.50
        --- Total Charges <= 0.01
            --- Monthly Charges <= 0.51
                --- class: 1
            --- Monthly Charges > 0.51
                --- Total Charges <= 0.01
                    --- class: 1
                --- Total Charges > 0.01
                    --- class: 1
        --- Total Charges > 0.01
            --- Dependents_Yes <= 0.50
                --- Monthly Charges <= 0.60
--- Tenure Months > 0.22
    --- Internet Service_Fiber optic <= 0.50
        --- Monthly Charges <= 0.06
            --- Monthly Charges <= 0.01
                --- class: 0
            --- Monthly Charges > 0.01
                --- class: 0
        --- Monthly Charges > 0.06
            --- Total Charges <= 0.37
                --- Dependents_Yes <= 0.50
                    --- class: 0
                --- Dependents_Yes > 0.50
                    --- class: 0
            --- Total Charges > 0.37
                --- Dependents_Yes <= 0.50
                    --- class: 0
                --- Dependents_Yes > 0.50
                    --- class: 0
    --- Internet Service_Fiber optic > 0.50
        --- Dependents_Yes <= 0.50
            --- Tenure Months <= 0.60
                --- Payment Method_Electronic check <= 0.50
                    --- class: 0
                --- Payment Method_Electronic check > 0.50
                    --- class: 1
            --- Tenure Months > 0.60
                --- Tenure Months <= 0.98
                    --- class: 0
                --- Tenure Months > 0.98
                    --- class: 0

```

Figure Texte export de DT avec RFE

3.2 Modele LogisticRegression (LR)

```

6]: from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

model = LogisticRegression()
rfe = RFE(model, n_features_to_select=9)
rfe.fit(x_train, y_train)
print("Sélection de variables", rfe.support_)
print("Classement de variables", rfe.ranking_)
print("Variables sélectionnées :", list(x.columns[rfe.support_]))

Sélection de variables [False False False  True False  True False False False  True
 False False False False False False  True  True  True False]
Classement de variables [17 18 13 1 12 1 9 1 6 3 2 1 15 20 16 14 4 11 7 19 5 1 1 10
 21 1 22 1 8 1]
Variables sélectionnées : ['Dependents_Yes', 'Multiple Lines_No phone service', 'Internet Service_Fiber optic', 'Online Backup_No internet service', 'Contract_One year', 'Contract_Two year', 'Payment Method_Electronic check', 'Tenure Months', 'Total Charges']

7]: x_train = rfe.transform(x_train)
x_test = rfe.transform(x_test)
print(x_train.shape)
print(x_test.shape)

(4922, 9)
(2110, 9)

```

```

: from sklearn.linear_model import LogisticRegression

logre_model = LogisticRegression(random_state=0, C=1.414141, penalty = 'l2', solver='liblinear')
logre_model.fit(x_train, y_train)

print('train score : ', logre_model.score(x_train, y_train) )
print('test score : ', logre_model.score(x_test,y_test) )

train score :  0.8057700121901666
test score :  0.8061611374407583

```

Figure : Score de RL avec RFE

3.2.1 Matrice de confusion du LR

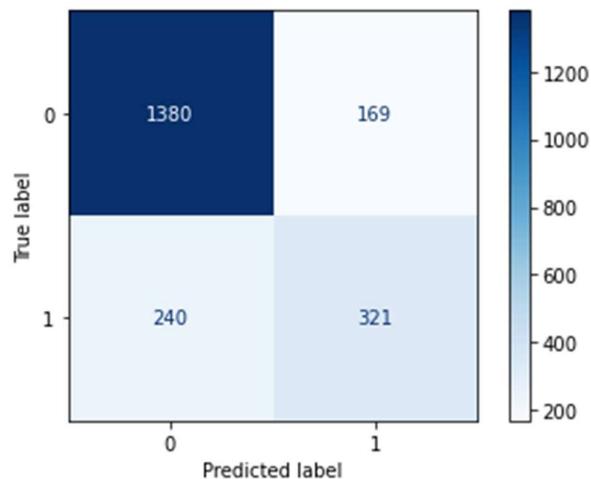


Figure : Matrice de confusion de LR avec RFE

3.2.2 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85185 | 0.89090 | 0.87094 | 1549 |
| 1 | 0.65510 | 0.57219 | 0.61085 | 561 |
| accuracy | | | 0.80616 | 2110 |
| macro avg | 0.75348 | 0.73154 | 0.74089 | 2110 |
| weighted avg | 0.79954 | 0.80616 | 0.80179 | 2110 |

Figure : Rapport de classification de LR avec RFE

3.2.3 Courbe Roc du LR

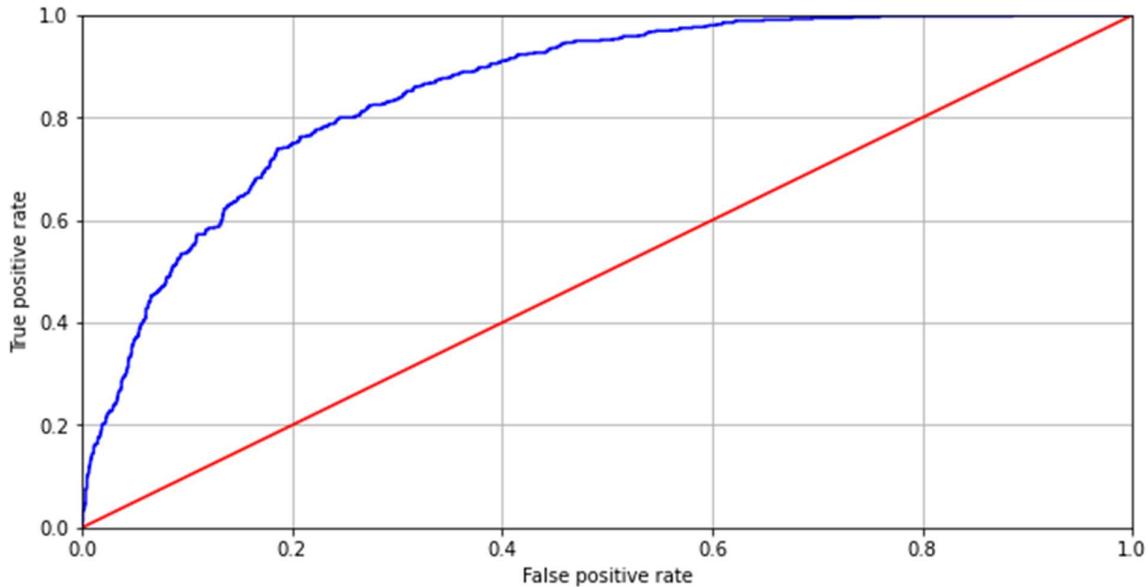


Figure : Courbe ROC de LR avec RFE

3.3 Modele RandomForestClassifier (RFC)

```

: from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE

model = RandomForestClassifier()
rfe = RFE(model, n_features_to_select=9)
rfe.fit(x_train, y_train)
print("Sélection de variables", rfe.support_)
print("Classement de variables", rfe.ranking_)
print("Variables sélectionnées : ", list(x.columns[rfe.support_]))

Sélection de variables [ True False False  True False False False  True False False False
 False False False False False False False  True  True  True False
 False  True False  True  True  True]
Classement de variables [ 1  6  3  1 16 21  8  1 18 20  5 11  7 15  9 22  4 17 12 19 10  1  1  2
 14  1 13  1  1  1]
Variables sélectionnées : ['Gender_Male', 'Dependents_Yes', 'Internet Service_Fiber optic', 'Contract_One year', 'Contract_Two year', 'Payment Method_Electronic check', 'Tenure Months', 'Monthly Charges', 'Total Charges']

: x_train = rfe.transform(x_train)
x_test = rfe.transform(x_test)
print(x_train.shape)
print(x_test.shape)

(4922, 9)
(2110, 9)

```



```

: from sklearn.ensemble import RandomForestClassifier

random_model=RandomForestClassifier(criterion='gini', max_depth=7,random_state=98)
random_model.fit(x_train,y_train)

print('train score : ',random_model.score(x_train,y_train))
print('test score : ',random_model.score(x_test,y_test))

train score :  0.8234457537586347
test score :  0.8066350710900474

```

Figure : Score de RFC avec RFE

3.3.1 Matrice de confusion du RFC

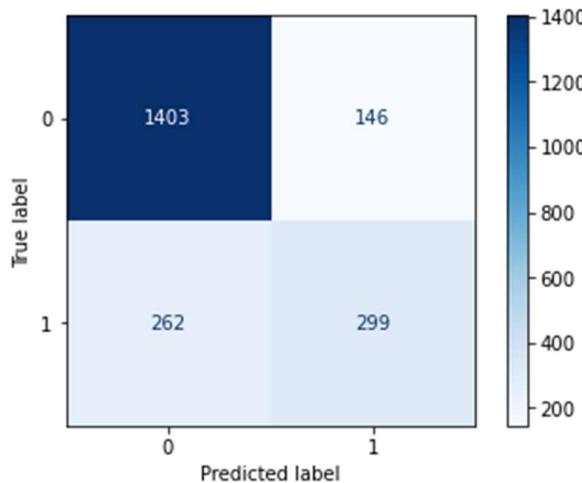


Figure : Matrice de confusion de RFC avec RFE

3.3.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84264 | 0.90575 | 0.87306 | 1549 |
| 1 | 0.67191 | 0.53298 | 0.59443 | 561 |
| accuracy | | | | |
| macro avg | 0.75728 | 0.71936 | 0.73374 | 2110 |
| weighted avg | 0.79725 | 0.80664 | 0.79898 | 2110 |

Figure : Rapport de classification de RFC avec RFE

3.3.3 Courbe Roc du RFC

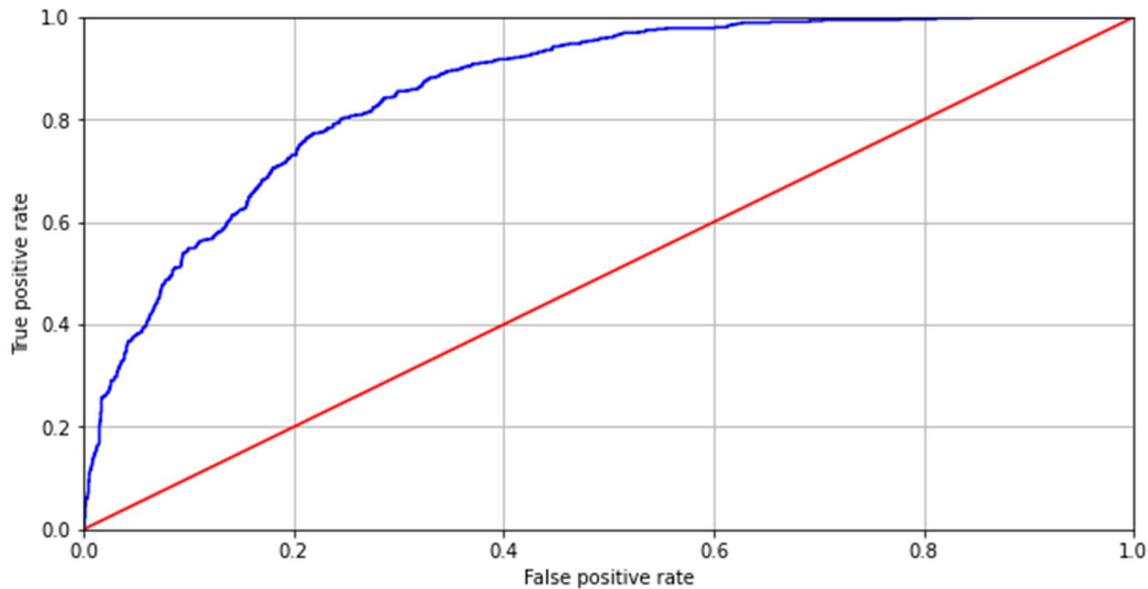


Figure : Courbe de ROC de RFC avec RFE

3.4 Modele XGboost(xgb)

```
: from xgboost import XGBClassifier
from sklearn.feature_selection import RFE

model = XGBClassifier()
rfe = RFE(model, n_features_to_select=9)
rfe.fit(x_train, y_train)
print("Sélection de variables", rfe.support_)
print("Classement de variables", rfe.ranking_)
print("Variables sélectionnées :", list(x.columns[rfe.support_]))

Sélection de variables [False False False True True False False True True False False
False False False False False False False True True True False
False True False True False False]
Classement de variables [11 8 7 1 1 18 10 1 1 16 4 17 9 20 13 19 3 21 12 22 1 1 1 2
15 1 14 1 6 5]
Variables sélectionnées : ['Dependents_Yes', 'Phone Service_Yes', 'Internet Service_Fiber optic', 'Internet Service_No', 'Streaming Movies_Yes', 'Contract_One year', 'Contract_Two year', 'Payment Method_Electronic check', 'Tenure Months']

2]: x_train = rfe.transform(x_train)
x_test = rfe.transform(x_test)
print(x_train.shape)
print(x_test.shape)

(5625, 9)
(1407, 9)

: from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=3,n_estimators=180,learning_rate=0.1)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

[00:42:44] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
0.8094222222222223
0.8045486851457001
```

Figure : Score de XGB avec RFE

3.4.1 Matrice de confusion du xgb

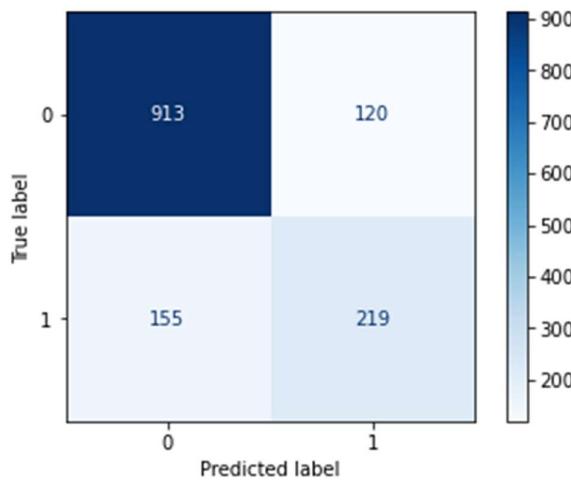


Figure : Matrice de confusion de XGB avec RFE

3.4.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85487 | 0.88383 | 0.86911 | 1033 |
| 1 | 0.64602 | 0.58556 | 0.61431 | 374 |
| accuracy | | | 0.80455 | 1407 |
| macro avg | 0.75044 | 0.73470 | 0.74171 | 1407 |
| weighted avg | 0.79935 | 0.80455 | 0.80138 | 1407 |

Figure : Rapport de classification de XGB avec RFE

3.4.3 Courbe Roc du xgb

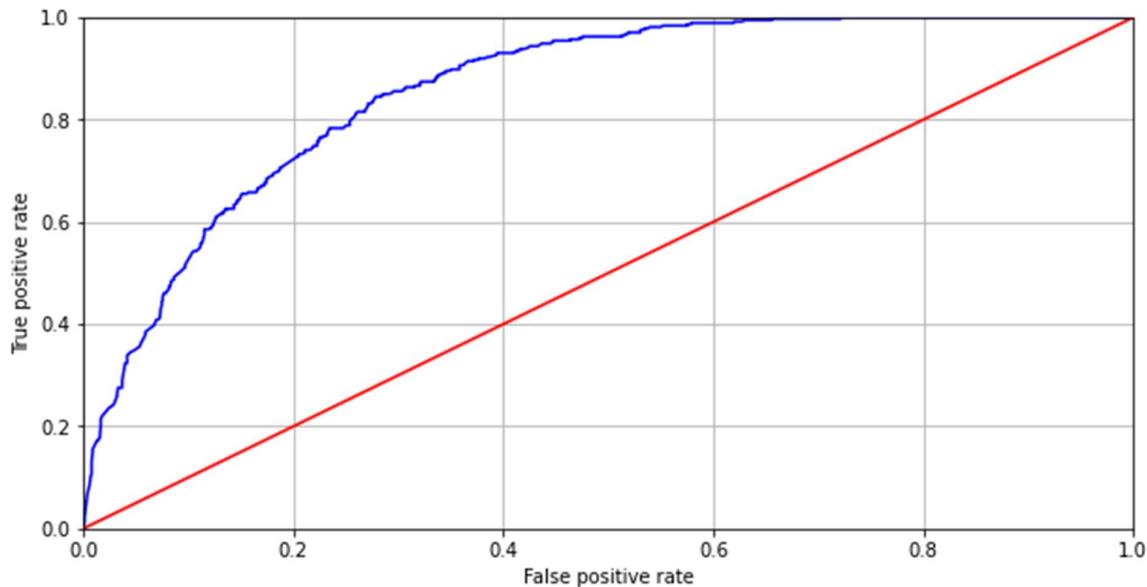


Figure : Courbe de ROC de XGB avec RFE

3.5 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|------------------------|-------------|------------|-----------------|------------------|---------|--------|
| DecisionTreeClassifier | 0.8051 | 0.7995 | 0.8652 | 0.6086 | 0.7995 | 0.8359 |
| LogisticRegression | 0.8057 | 0.8061 | 0.8709 | 0.6108 | 0.8061 | 0.8548 |
| RandomForestClassifier | 0.8234 | 0.8066 | 0.8730 | 0.5944 | 0.8066 | 0.8575 |
| XGBClassifier | 0.8094 | 0.8045 | 0.8691 | 0.6143 | 0.8045 | 0.8583 |

Recursive Feature Elimination (RFE) (with 9 features)

Figure : Table de résumé des modèles avec RFE

➔ Le modèle RandomForestClassifier donne les meilleurs résultats si on utilise RFE pour la sélection des variables (avec 9 variables restantes)

4 IV Application des modeles en utilisant la corrélation avec la variable cible

4.1 Modele KNeighbors Classifier (KNN)

```
from sklearn.neighbors import KNeighborsClassifier  
  
knn_model=KNeighborsClassifier(n_neighbors=49,metric='euclidean')  
  
knn_model.fit(x_train,y_train)  
  
print('score du train ',knn_model.score(x_train,y_train))  
print('score du test' ,knn_model.score(x_test,y_test))  
  
score du train  0.8049573344169036  
score du test 0.7995260663507109
```

Figure : Score de KNN avec corrélation avec variable cible

4.1.1 Matrice de confusion du KNN

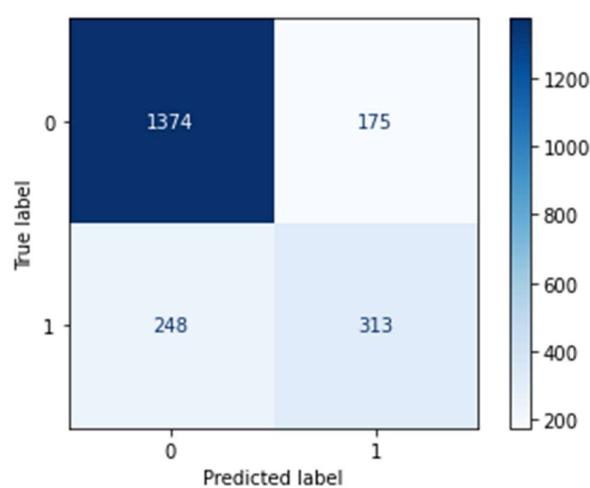


Figure : Matrice de confusion de kNN avec corrélation avec variable cible

4.1.2 Rapport de classification du KNN

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|----------|
| 0 | 0.847102 | 0.887024 | 0.866604 | 1549 |
| 1 | 0.641393 | 0.557932 | 0.596759 | 561 |
| accuracy | | | | 0.799526 |
| macro avg | 0.744248 | 0.722478 | 0.731681 | 2110 |
| weighted avg | 0.792409 | 0.799526 | 0.794858 | 2110 |

Figure : Rapport de classification de KNN avec corrélation avec variable cible

4.1.3 Courbe Roc du KNN

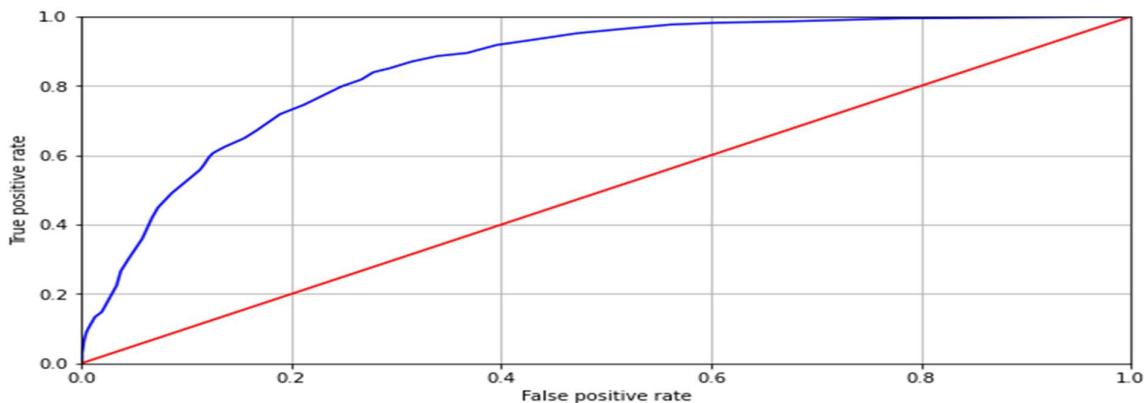


Figure : Courbe ROC de KNN avec corrélation avec variable cible

4.2 Modele DecisionTreeClassifier(DT)

```
: from sklearn.tree import DecisionTreeClassifier

tree_model=DecisionTreeClassifier(criterion='gini',max_depth=6,random_state=0)
tree_model.fit(x_train,y_train)
print('train score : ', tree_model.score(x_train,y_train))
print('test score : ', tree_model.score(x_test,y_test))

train score :  0.8082080455099553
test score :  0.79478672985782
```

Figure : Score de DT avec corrélation avec variable cible

4.2.1 Matrice de confusion de DT

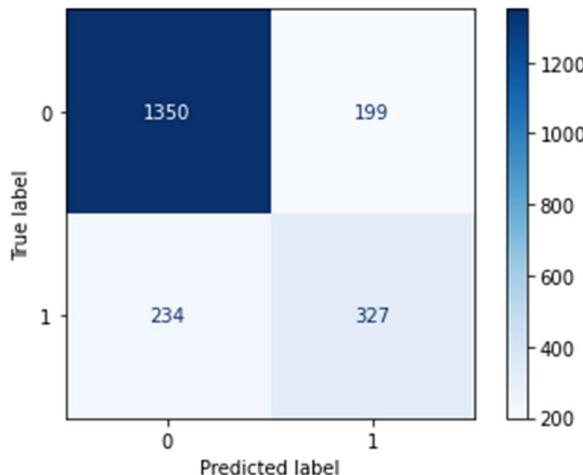


Figure : Matrice de confusion de DT avec corrélation avec variable cible

4.2.2 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.852273 | 0.871530 | 0.861794 | 1549 |
| 1 | 0.621673 | 0.582888 | 0.601656 | 561 |
| accuracy | | | 0.794787 | 2110 |
| macro avg | 0.736973 | 0.727209 | 0.731725 | 2110 |
| weighted avg | 0.790962 | 0.794787 | 0.792629 | 2110 |

Figure : Rapport de classification de DT avec corrélation avec variable cible

4.2.3 Courbe Roc du DT

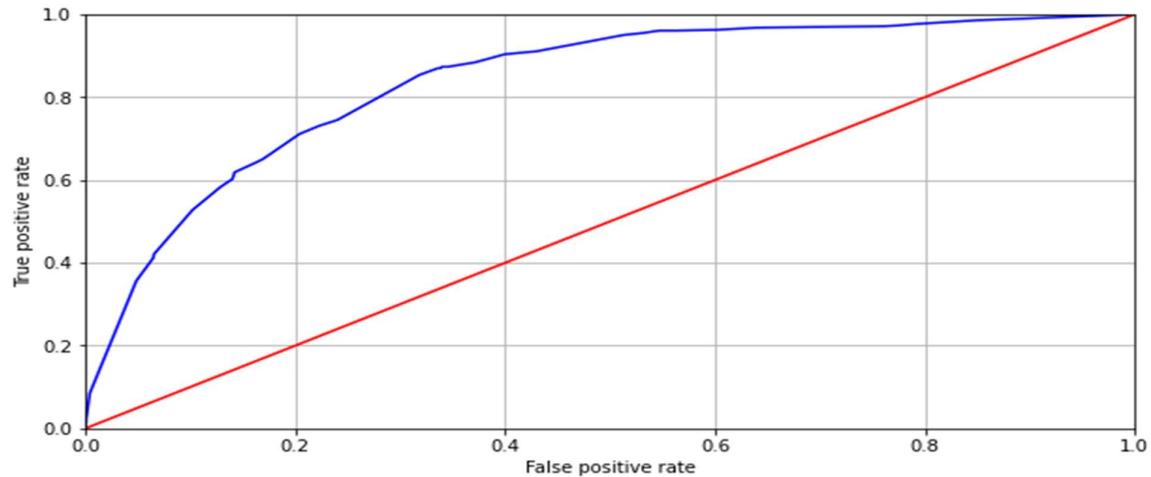


Figure : Courbe ROC de DT avec corrélation avec variable cible

4.2.4 Arbre de Decision du DT

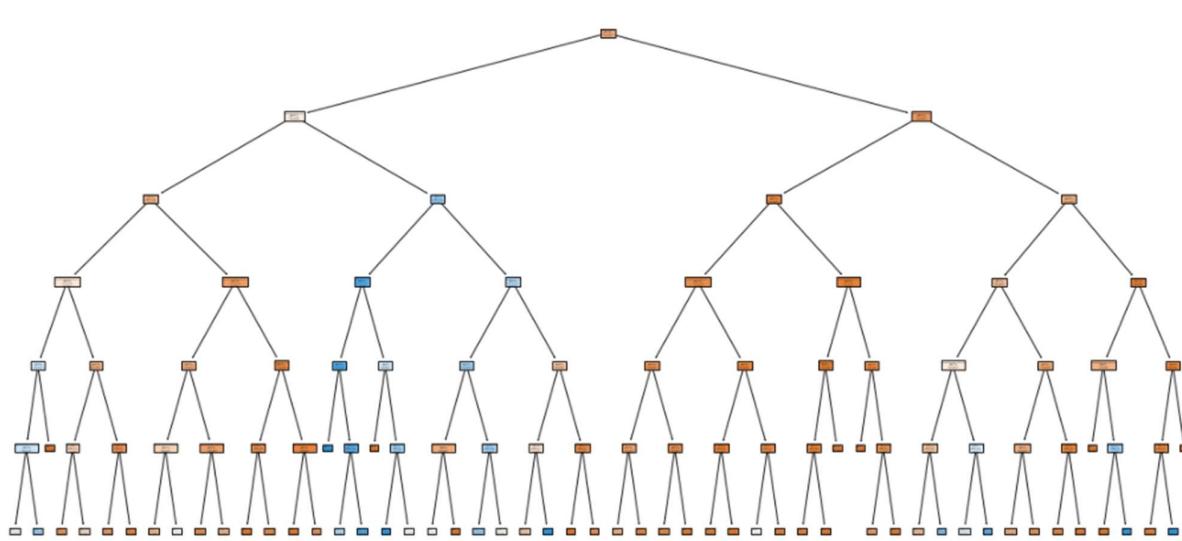


Figure : Arbre de décision avec corrélation avec variable cible

4.2.5 Text d'export du DT

```

--- tenure Months <= 0.22
    --- Internet Service_Fiber optic <= 0.50
        --- Tenure Months <= 0.04
            --- Device Protection_No internet service <= 0.50
                --- Total Charges <= 0.02
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 1
                    --- Payment Method_Electronic check > 0.50
                        --- class: 1
                --- Total Charges > 0.02
                    --- class: 0
            --- Device Protection_No internet service > 0.50
                --- Total Charges <= 0.00
                    --- Total Charges <= 0.00
                        --- class: 0
                    --- Total Charges > 0.00
                        --- class: 0
                --- Total Charges > 0.00
                    --- Total Charges <= 0.00
                        --- class: 0
                    --- Total Charges > 0.00
                        --- class: 0
        --- Tenure Months > 0.04
            --- Device Protection_No internet service <= 0.50
                --- Total Charges <= 0.04
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 0
                    --- Payment Method_Electronic check > 0.50
                        --- class: 0
                --- Total Charges > 0.04
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 0
                    --- Payment Method_Electronic check > 0.50
                        --- class: 0
            --- Device Protection_No internet service > 0.50
                --- Dependents_Yes <= 0.50
--- Internet Service_Fiber optic > 0.50
    --- Total Charges <= 0.01
        --- Dependents_Yes <= 0.50
            --- Total Charges <= 0.01
                --- class: 1
            --- Total Charges > 0.01
                --- Total Charges <= 0.01
                    --- class: 1
                --- Total Charges > 0.01
                    --- class: 1
        --- Dependents_Yes > 0.50
            --- Total Charges <= 0.01
                --- class: 0
            --- Total Charges > 0.01
                --- Total Charges <= 0.01
                    --- class: 1
                --- Total Charges > 0.01
                    --- class: 0
            --- Total Charges > 0.01
                --- Dependents_Yes <= 0.50
                    --- Total Charges <= 0.01
                        --- Payment Method_Electronic check <= 0.50
                            --- class: 0
                        --- Payment Method_Electronic check > 0.50
                            --- class: 0
                    --- Total Charges > 0.01
                        --- Tenure Months <= 0.20
                            --- class: 1
                        --- Tenure Months > 0.20
                            --- class: 0
                --- Dependents_Yes > 0.50
                    --- Total Charges <= 0.13
                        --- Total Charges <= 0.12
                            --- class: 0
                        --- Total Charges > 0.12

```

Figure : Texte d'export DT avec corrélation avec variable cible

4.3 Modele NaiveBayes (NB)

```

]: from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
      'bernoulli': BernoulliNB(),
      'multinomial': MultinomialNB()}
scores = {}
for key, model in nb.items():  #nb.items; parcourir cle et valeur
    s = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)
scores
]: {'gaussian': 0.6436292352771243,
 'bernoulli': 0.7192105154554084,
 'multinomial': 0.7212430357806117}

]: bayes_model = MultinomialNB()
bayes_model.fit(x_train, y_train)
print('train score : ', bayes_model.score(x_train,y_train))
print('test score : ', bayes_model.score(x_test,y_test))

train score :  0.719626168224299
test score :  0.7322274881516587

```

Figure : Score de NB avec corrélation avec variable cible

4.3.1 Matrice de confusion du NB

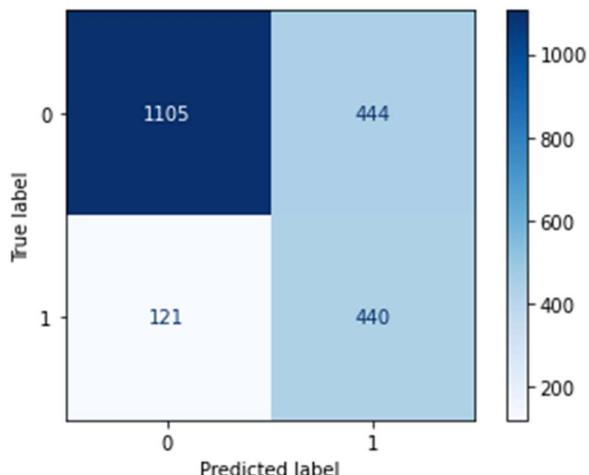


Figure : Matrice de confusion deNB avec corrélation avec variable cible

4.3.2 Rapport de classification du NB

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.90131 | 0.71336 | 0.79640 | 1549 |
| 1 | 0.49774 | 0.78431 | 0.60900 | 561 |
| accuracy | | | 0.73223 | 2110 |
| macro avg | 0.69952 | 0.74884 | 0.70270 | 2110 |
| weighted avg | 0.79401 | 0.73223 | 0.74657 | 2110 |

4.3.3 Courbe Roc du NB

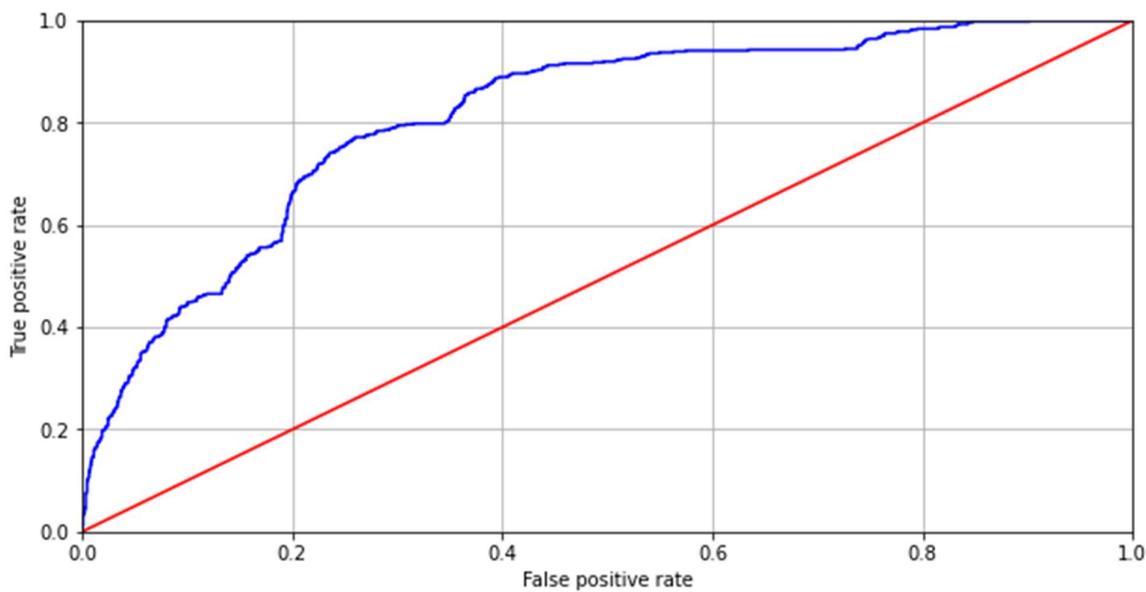


Figure : Courbe ROC de NB avec corrélation avec variable cible

4.4 Modele LogisticRegression (LR)

```
: from sklearn.linear_model import LogisticRegression

logre_model = LogisticRegression(random_state=0, C=0.10101, penalty = 'l2', solver='lbfgs')
logre_model.fit(x_train, y_train)

print('train score : ' , logre_model.score(x_train, y_train) )
print('test score : ' , logre_model.score(x_test,y_test) )

train score :  0.8033319788703779
test score :  0.8056872037914692
```

Figure : Score de LR avec corrélation avec variable cible

4.4.1 Matrice de confusion du LR

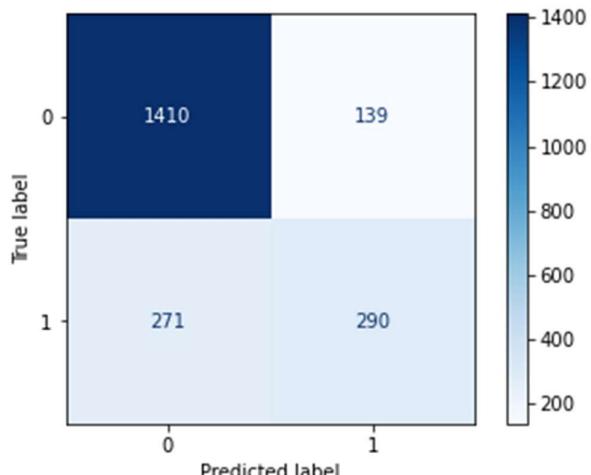


Figure : Matrice de confusion de LR avec corrélation avec variable cible

4.4.2 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83879 | 0.91026 | 0.87307 | 1549 |
| 1 | 0.67599 | 0.51693 | 0.58586 | 561 |
| accuracy | | | 0.80569 | 2110 |
| macro avg | 0.75739 | 0.71360 | 0.72946 | 2110 |
| weighted avg | 0.79550 | 0.80569 | 0.79670 | 2110 |

Figure : Rapport de classification de LR avec corrélation avec variable cible

4.4.3 Courbe Roc du LR

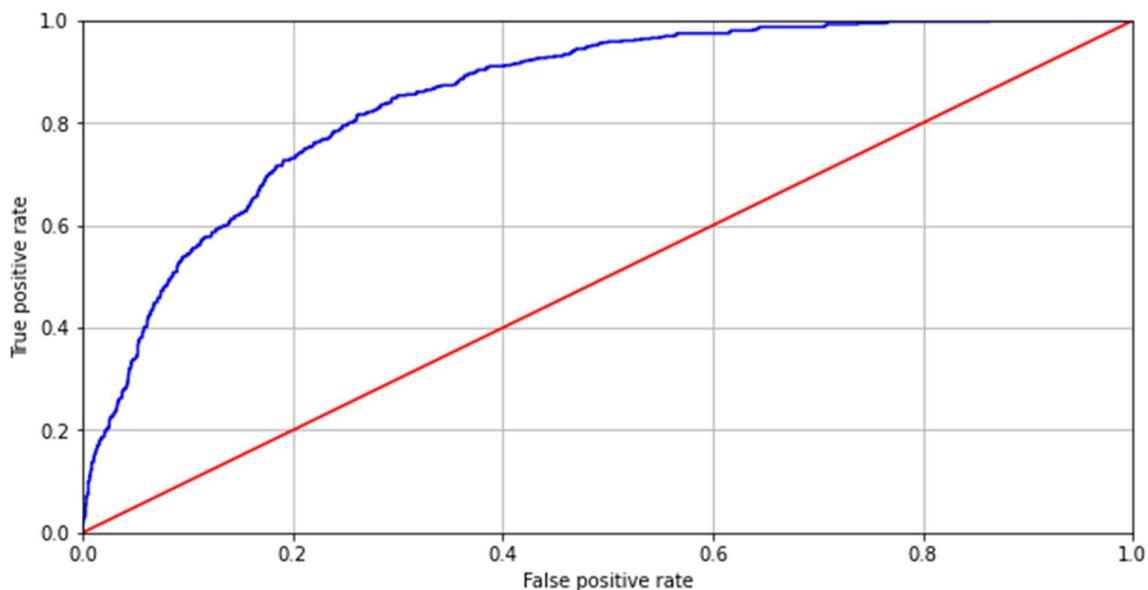


Figure : Courbe ROC de LR avec corrélation avec variable cible

4.5 Modele RandomForestClassifier (RFC)

```
from sklearn.ensemble import RandomForestClassifier  
  
random_model=RandomForestClassifier(criterion='gini', max_depth=5,random_state=22)  
random_model.fit(x_train,y_train)  
  
print('train score : ',random_model.score(x_train,y_train))  
print('test score : ',random_model.score(x_test,y_test))
```

```
train score :  0.8029256399837464  
test score :  0.8
```

Figure : Score de RFC avec corrélation avec variable cible

4.5.1 Matrice de confusion du RFC

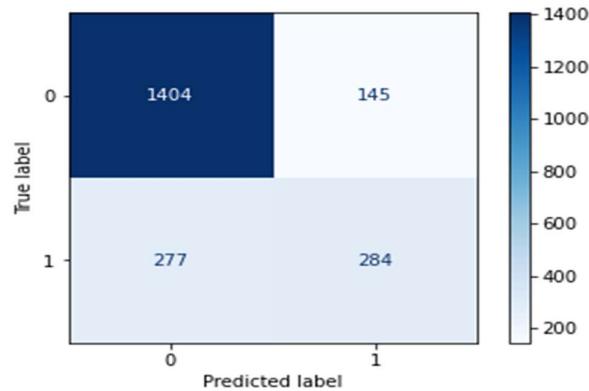


Figure : Matrice de confusion RFC avec corrélation avec variable cible

4.5.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83522 | 0.90639 | 0.86935 | 1549 |
| 1 | 0.66200 | 0.50624 | 0.57374 | 561 |
| accuracy | | | 0.80000 | 2110 |
| macro avg | 0.74861 | 0.70632 | 0.72154 | 2110 |
| weighted avg | 0.78916 | 0.80000 | 0.79075 | 2110 |

Figure Rapport de classification RFC avec corrélation avec variable cible

4.5.3 Courbe Roc du RFC

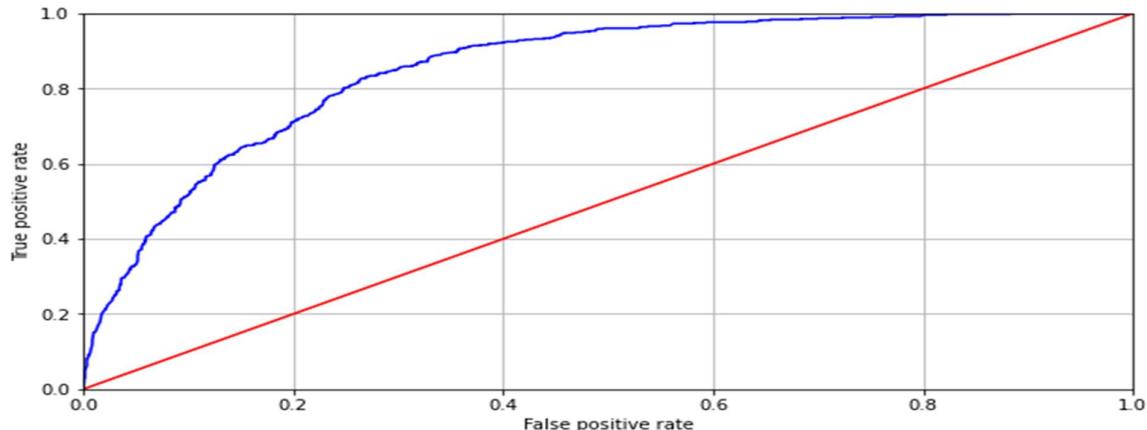


Figure : Courbe ROC de RFC avec corrélation avec variable cible

4.6 Modele XGboost(xgb)

```

from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=2,n_estimators=100,learning_rate=0.1)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

[23:32:40] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:147: The 'log' metric used with the objective 'binary:logistic' was changed from 'error' to 'logit_error'. Please restore the old behavior.
0.8058666666666666
0.8024164889836531

```

Figure : Score de XGB avec corrélation avec variable cible

4.6.1 Matrice de confusion du xgb

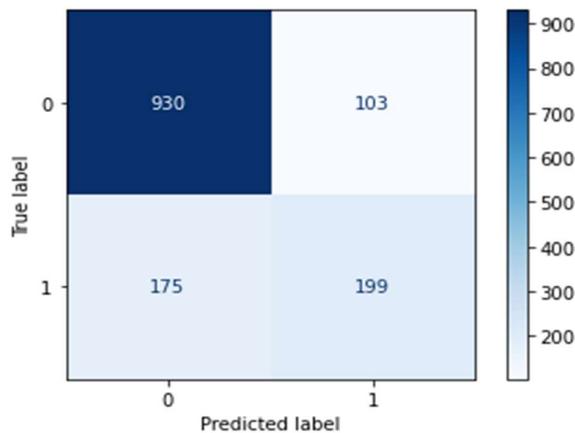


Figure : Matrice de confusion de XGB avec corrélation avec variable cible

4.6.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84163 | 0.90029 | 0.86997 | 1033 |
| 1 | 0.65894 | 0.53209 | 0.58876 | 374 |
| accuracy | | | 0.80242 | 1407 |
| macro avg | 0.75028 | 0.71619 | 0.72936 | 1407 |
| weighted avg | 0.79307 | 0.80242 | 0.79522 | 1407 |

Figure : Rapoort de classification de XGB avec corrélation avec variable cible

4.6.3 Courbe Roc du xgb

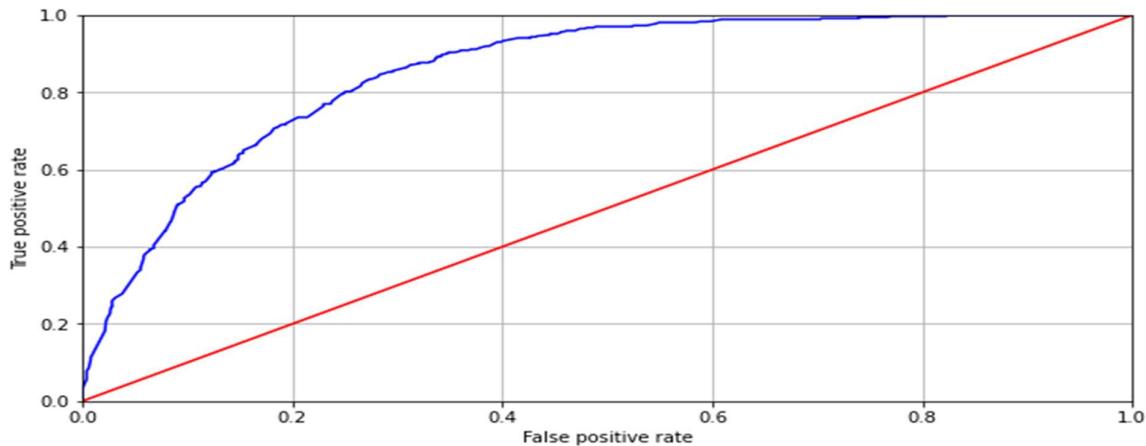


Figure : Courbe ROC de XGB avec corrélation avec variable cible

4.7 Modele Support Vector Machine (SVM)

```
: from sklearn import svm

svm_model=svm.SVC(C=0.1,kernel='rbf',gamma=1)
svm_model.fit(x_train,y_train)
print(svm_model.score(x_train,y_train))
print(svm_model.score(x_test,y_test))

0.7943111111111111
0.8024164889836531
```

Figure : Score de SVM avec corrélation avec variable cible

4.7.1 Matrice de confusion du SVM

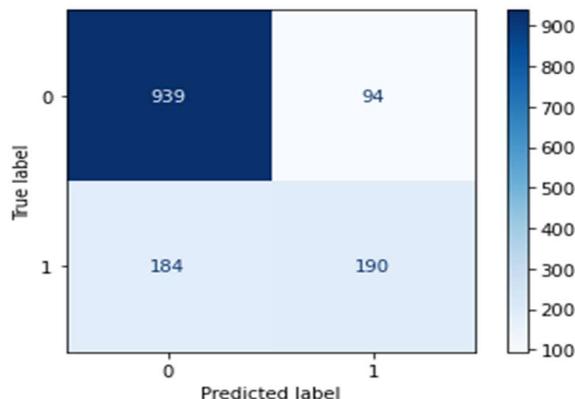


Figure : Matrice de confusion de SVM avec corrélation avec variable cible

4.7.2 Rapport de classification du SVM

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83615 | 0.90900 | 0.87106 | 1033 |
| 1 | 0.66901 | 0.50802 | 0.57751 | 374 |
| accuracy | | | 0.80242 | 1407 |
| macro avg | 0.75258 | 0.70851 | 0.72428 | 1407 |
| weighted avg | 0.79173 | 0.80242 | 0.79303 | 1407 |

Figure : Rapport de classification de SVM avec corrélation avec variable cible

4.8 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8049 | 0.7995 | 0.8666 | 0.5967 | 0.7995 | 0.8827 |
| DecisionTreeClassifier | 0.8082 | 0.7947 | 0.8617 | 0.6016 | 0.7947 | 0.8381 |
| NaiveBayes(MultiNomial) | 0.7196 | 0.7322 | 0.7964 | 0.6090 | 0.7322 | 0.8131 |
| LogisticRegression | 0.8033 | 0.8056 | 0.8730 | 0.5858 | 0.8056 | 0.8521 |
| RandomForestClassifier | 0.8029 | 0.8000 | 0.8693 | 0.5737 | 0.8000 | 0.8522 |
| XGBClassifier | 0.8058 | 0.8024 | 0.8699 | 0.5887 | 0.8024 | 0.8561 |
| SVM | 0.7943 | 0.8024 | 0.8710 | 0.5775 | 0.8024 | False |

Correlation between features and target (with 13 features)

Figure : Table de résumé des modèles avec corrélation avec variable cible

- ➔ Le modèle XGBClassifier donne les meilleurs résultats si on utilise la corrélation avec la variable cible pour la sélection des variables (avec 13 variables restantes)

5 Application des modeles en utilisant la corrélation entre les variables

5.1 Modele KNeighbors Classifier (KNN)

```
: from sklearn.neighbors import KNeighborsClassifier  
  
knn_model=KNeighborsClassifier(n_neighbors=35,metric='euclidean')  
  
knn_model.fit(x_train,y_train)  
  
print('score du train ',knn_model.score(x_train,y_train))  
print('score du test' ,knn_model.score(x_test,y_test))  
  
score du train  0.8086143843965867  
score du test 0.7985781990521327
```

Figure : Score de KNN avec corrélation entre les variables

5.1.1 Matrice de confusion du KNN

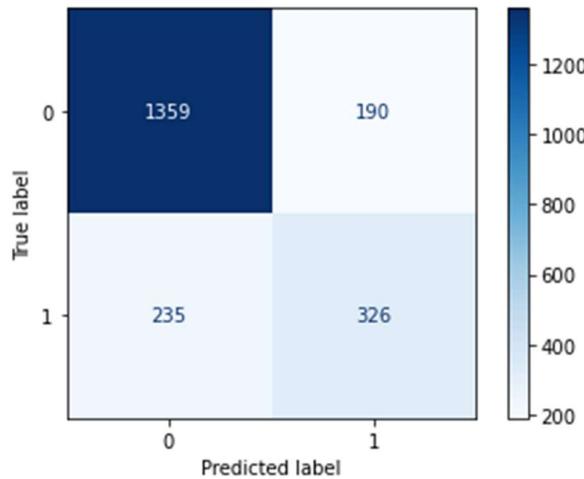


Figure : Matrice de confusion de kNN avec corrélation entre variables

5.1.2 Rapport de classification du KNN

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.852572 | 0.877340 | 0.864779 | 1549 |
| 1 | 0.631783 | 0.581105 | 0.605385 | 561 |
| accuracy | | | 0.798578 | 2110 |
| macro avg | 0.742178 | 0.729223 | 0.735082 | 2110 |
| weighted avg | 0.793869 | 0.798578 | 0.795812 | 2110 |

Figure : Rapport de classification de kNN avec corrélation entre variables

5.1.3 Courbe Roc du KNN

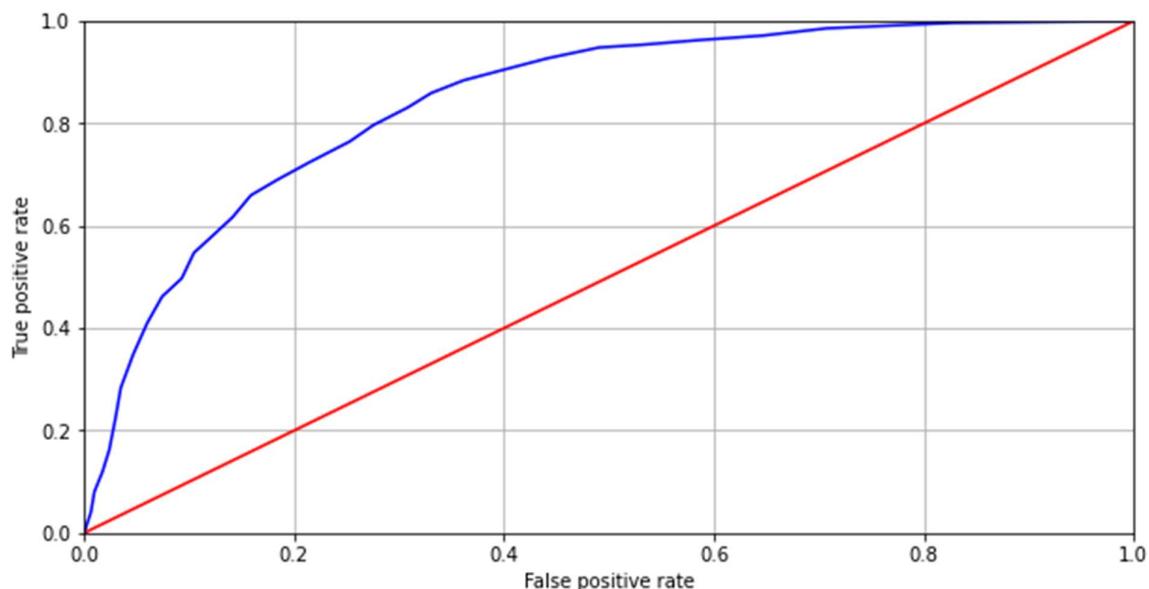


Figure : Courbe ROC de kNN avec corrélation entre variables

5.2 Modele DecisionTreeClassifier(DT)

```
from sklearn.tree import DecisionTreeClassifier

tree_model=DecisionTreeClassifier(criterion='gini',max_depth=7,random_state=0)
tree_model.fit(x_train,y_train)
print('train score : ', tree_model.score(x_train,y_train))
print('test score : ', tree_model.score(x_test,y_test))

train score :  0.8216172287687932
test score :  0.7914691943127962
```

Figure : Score de DT avec corrélation entre les variables

5.2.1 Matrice de confusion de DT

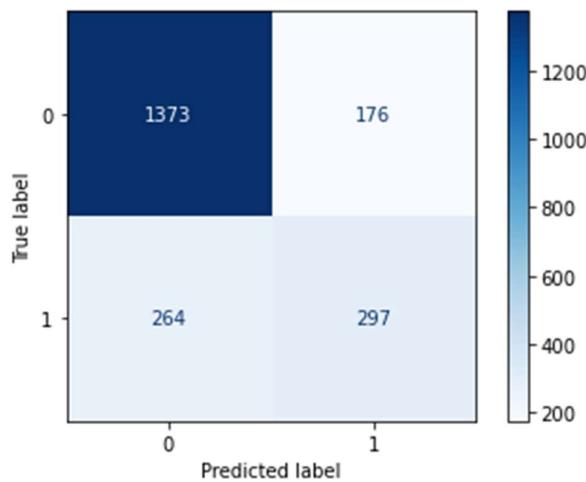


Figure : Matrice de confusion de DT avec corrélation entre variables

5.2.3 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|---------|
| 0 | 0.838729 | 0.886378 | 0.861896 | 1549 |
| 1 | 0.627907 | 0.529412 | 0.574468 | 561 |
| accuracy | | | 0.791469 | 2110 |
| macro avg | 0.733318 | 0.707895 | 0.718182 | 2110 |
| weighted avg | 0.782677 | 0.791469 | 0.785475 | 2110 |

Figure : Rapport de classification de DT avec corrélation entre variables

5.2.4 Courbe Roc du DT

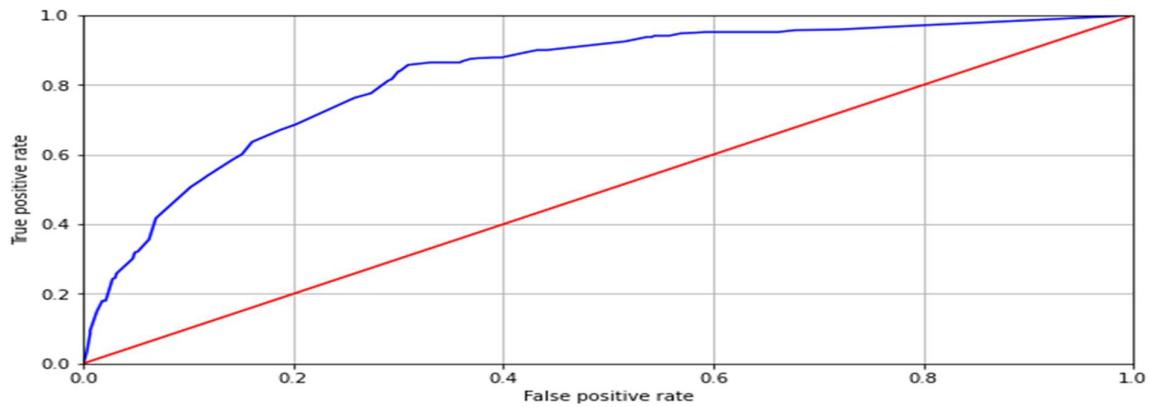


Figure : Courbe ROC de DT avec corrélation entre variables

5.2.5 Arbre de Decision du DT

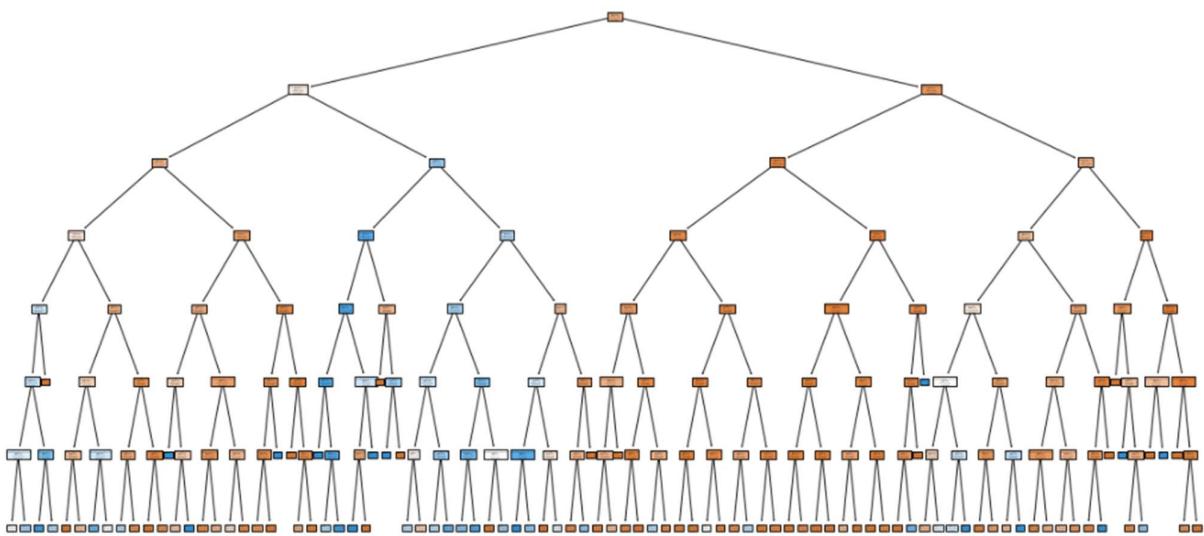


Figure : Arbre de décision de DT avec corrélation entre variables

5.2.6 Text d'export du DT

```

--- Tenure Months <= 0.22
    --- Internet Service_Fiber optic <= 0.50
        |--- Tenure Months <= 0.04
            |--- Internet Service_No <= 0.50
                |--- Total Charges <= 0.02
                    |--- Senior Citizen_Yes <= 0.50
                        |--- Payment Method_Electronic check <= 0.50
                            |--- class: 0
                        |--- Payment Method_Electronic check > 0.50
                            |--- class: 1
                    |--- Senior Citizen_Yes > 0.50
                        |--- Phone Service_Yes <= 0.50
                            |--- class: 1
                        |--- Phone Service_Yes > 0.50
                            |--- class: 1
                |--- Total Charges > 0.02
                    |--- class: 0
    |--- Internet Service_No > 0.50
        --- Total Charges <= 0.00
            |--- Paperless Billing_Yes <= 0.50
                |--- Monthly Charges <= 0.01
                    |--- class: 0
                |--- Monthly Charges > 0.01
                    |--- class: 0
            |--- Paperless Billing_Yes > 0.50
                |--- Payment Method_Mailed check <= 0.50
                    |--- class: 1
                |--- Payment Method_Mailed check > 0.50
                    |--- class: 0
        |--- Total Charges > 0.00
            |--- Monthly Charges <= 0.01
                |--- Total Charges <= 0.00
                    |--- class: 1
                |--- Total Charges > 0.00
                    |--- class: 0
            |--- Monthly Charges > 0.01
                |--- Internet Service_No > 0.50
                    |--- Monthly Charges <= 0.02
                        |--- Total Charges <= 0.04
                            |--- Dependents_Yes <= 0.50
                                |--- class: 0
                            |--- Dependents_Yes > 0.50
                                |--- class: 0
                        |--- Total Charges > 0.04
                            |--- class: 1
                    |--- Monthly Charges > 0.02
                        |--- Monthly Charges <= 0.03
                            |--- class: 0
                        |--- Monthly Charges > 0.03
                            |--- Total Charges <= 0.01
                                |--- class: 0
                            |--- Total Charges > 0.01
                                |--- class: 0
                |--- Internet Service_Fiber optic > 0.50
                    |--- Total Charges <= 0.01
                        |--- Tech Support_Yes <= 0.50
                            |--- Dependents_Yes <= 0.50
                                |--- Total Charges <= 0.01
                                    |--- class: 1
                                |--- Total Charges > 0.01
                                    |--- Total Charges <= 0.01
                                        |--- class: 1
                                        |--- Total Charges > 0.01
                                            |--- class: 1
                                    |--- Dependents_Yes > 0.50
                                        |--- Payment Method_Mailed check <= 0.50
                                            |--- Partner_Yes <= 0.50
                                                |--- class: 1
                                            |--- Partner_Yes > 0.50
                                                |--- class: 0
                                        |--- Payment Method_Mailed check > 0.50
                                            |--- class: 1

```

Figure : Text d'export de DT avec corrélation entre variables

5.3 Modele NaiveBayes (NB)

```

: from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
      'bernoulli': BernoulliNB(),
      'multinomial': MultinomialNB()}
scores = {}
for key, model in nb.items(): #nb.items; parcourir cle et valeur
    s = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)
scores

: {'gaussian': 0.7626920061078784,
 'bernoulli': 0.7828040526598159,
 'multinomial': 0.7862605752961083}

: bayes_model = MultinomialNB()
bayes_model.fit(x_train, y_train)
print('train score : ', bayes_model.score(x_train,y_train))
print('test score : ', bayes_model.score(x_test,y_test))

train score :  0.7870784234051199
test score :  0.7985781990521327

```

Figure : Score de NB avec corrélation entre les variables

5.3.1 Matrice de confusion du NB

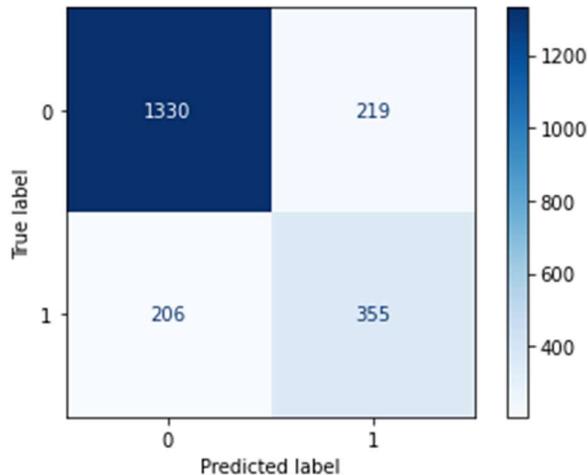


Figure : Matrice de confusion deNB avec corrélation entre variables

5.3.2 Rapport de classification du NB

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.86589 | 0.85862 | 0.86224 | 1549 |
| 1 | 0.61847 | 0.63280 | 0.62555 | 561 |
| accuracy | | | 0.79858 | 2110 |
| macro avg | 0.74218 | 0.74571 | 0.74389 | 2110 |
| weighted avg | 0.80010 | 0.79858 | 0.79931 | 2110 |

Figure : Rapport de classification de NB avec corrélation entre variables

5.3.3 Courbe Roc du NB

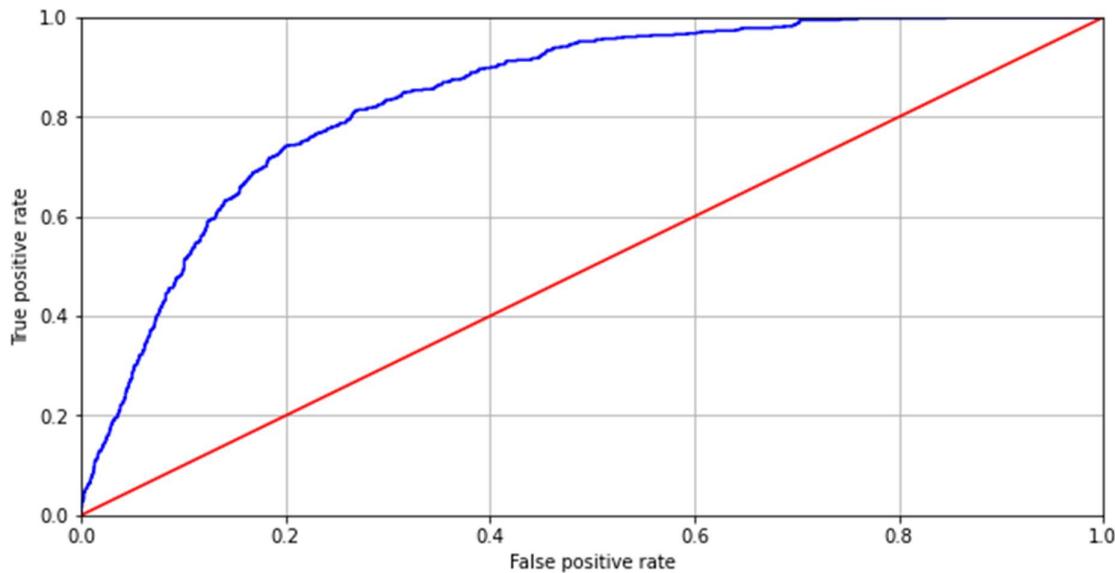


Figure : Courbe ROC de NB avec corrélation entre variables

5.4 Modèle LogisticRegression (LR)

```
: from sklearn.linear_model import LogisticRegression  
  
logre_model = LogisticRegression(random_state=0, C=1.31313, penalty = 'l1', solver='liblinear')  
logre_model.fit(x_train, y_train)  
  
print('train score : ' , logre_model.score(x_train, y_train) )  
print('test score : ' , logre_model.score(x_test,y_test) )  
  
train score :  0.8108492482730597  
test score :  0.8090047393364929
```

Figure : Score de LR avec corrélation entre les variables

5.3.2 Matrice de confusion du LR

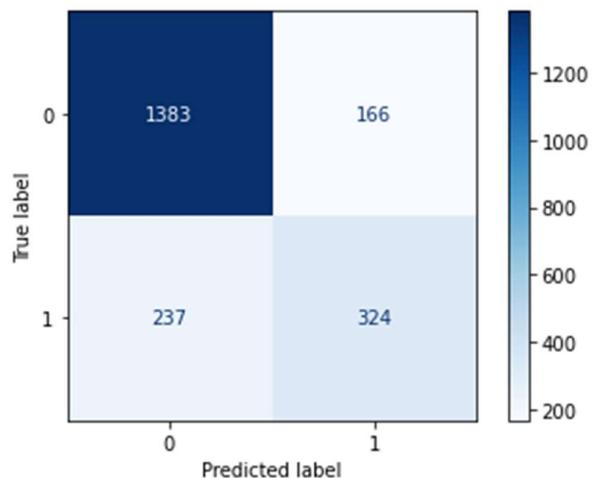


Figure : Matrice de confusion de LR avec corrélation entre variables

5.3.3 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85370 | 0.89283 | 0.87283 | 1549 |
| 1 | 0.66122 | 0.57754 | 0.61656 | 561 |
| accuracy | | | 0.80900 | 2110 |
| macro avg | 0.75746 | 0.73519 | 0.74469 | 2110 |
| weighted avg | 0.80253 | 0.80900 | 0.80469 | 2110 |

Figure Rapport de classification de LR avec corrélation entre variables

5.3.4 Courbe Roc du LR

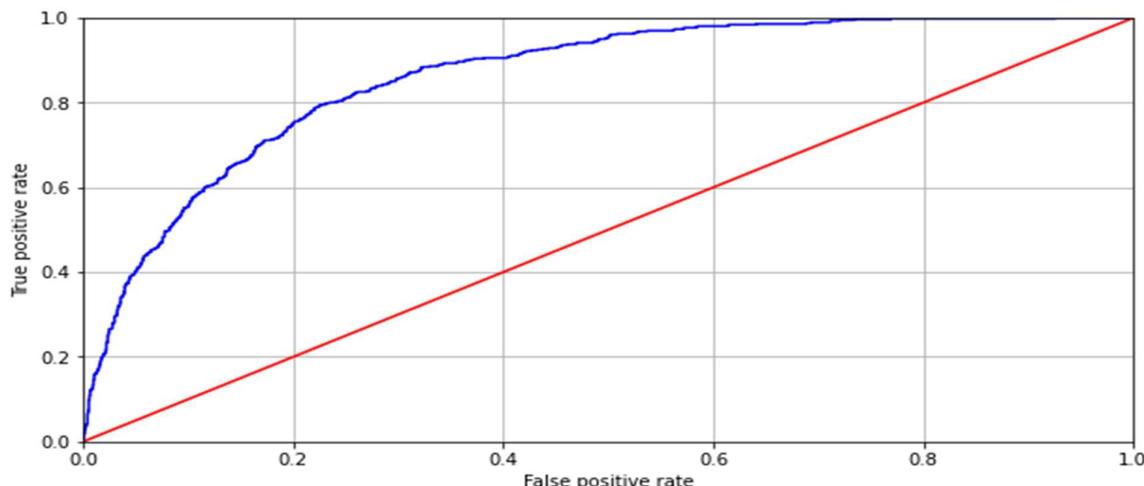


Figure : Courbe de ROC de LR avec corrélation entre variables

5.4 Modele RandomForestClassifier (RFC)

```
: from sklearn.ensemble import RandomForestClassifier  
  
random_model=RandomForestClassifier(criterion='gini', max_depth=9,random_state=81)  
random_model.fit(x_train,y_train)  
  
print('train score : ',random_model.score(x_train,y_train))  
print('test score : ',random_model.score(x_test,y_test))  
  
train score :  0.8519111111111111  
test score :  0.8137882018479033
```

Figure : Score de RFC avec corrélation entre les variables

5.4.1 Matrice de confusion du RFC

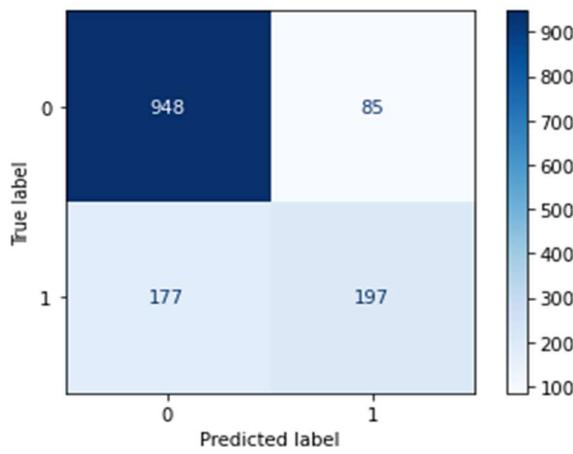


Figure : Matrice de confusion de RFC avec corrélation entre variables

5.4.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84267 | 0.91772 | 0.87859 | 1033 |
| 1 | 0.69858 | 0.52674 | 0.60061 | 374 |
| accuracy | | | 0.81379 | 1407 |
| macro avg | 0.77062 | 0.72223 | 0.73960 | 1407 |
| weighted avg | 0.80437 | 0.81379 | 0.80470 | 1407 |

Figure : Rapport de classification de RFC avec corrélation entre variables

5.4.3 Courbe Roc du RFC

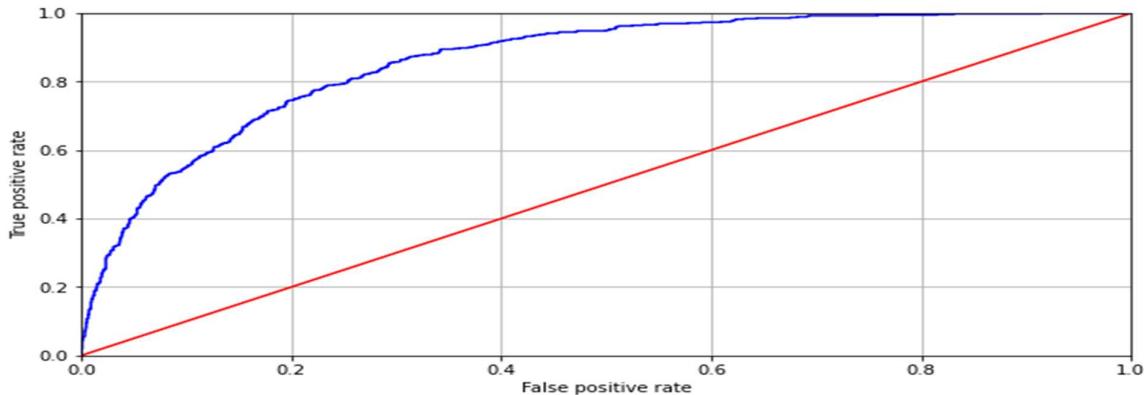


Figure : Courbe de ROC avec corrélation entre variables

5.5 Modele XGboost(xgb)

```

: from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=4,n_estimators=100,learning_rate=0.05)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

[23:51:49] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc
valuation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'
restore the old behavior.
0.8200888888888889
0.814498933901919

```

Figure : Score de XGB avec corrélation entre les variables

5.5.1 Matrice de confusion du xgb

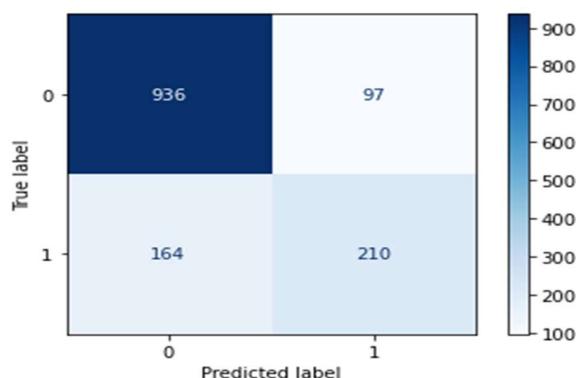


Figure : Matrice de confusion de XGB avec corrélation entre variables

5.5.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.85091 | 0.90610 | 0.87764 | 1033 |
| | 0.68404 | 0.56150 | 0.61674 | 374 |
| accuracy | | | | |
| macro avg | 0.76747 | 0.73380 | 0.74719 | 1407 |
| weighted avg | 0.80655 | 0.81450 | 0.80829 | 1407 |

Figure : Rapport de classification de XGB avec corrélation entre variables

5.5.3 Courbe Roc du xgb

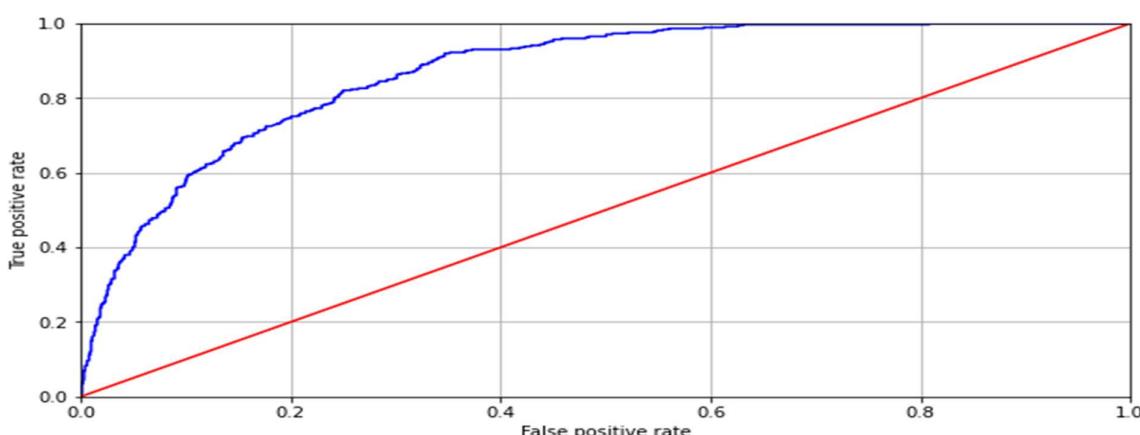


Figure Courbe ROC de XGB avec corrélation entre variables

5.6 Modele Support Vector Machine (SVM)

```
: from sklearn import svm

svm_model=svm.SVC(C=1,kernel='rbf',gamma=0.1)
svm_model.fit(x_train,y_train)
print(svm_model.score(x_train,y_train))
print(svm_model.score(x_test,y_test))

0.8186666666666667
0.8102345415778252
```

Figure : Score de SVM avec corrélation entre les variables

5.6.1 Matrice de confusion du SVM

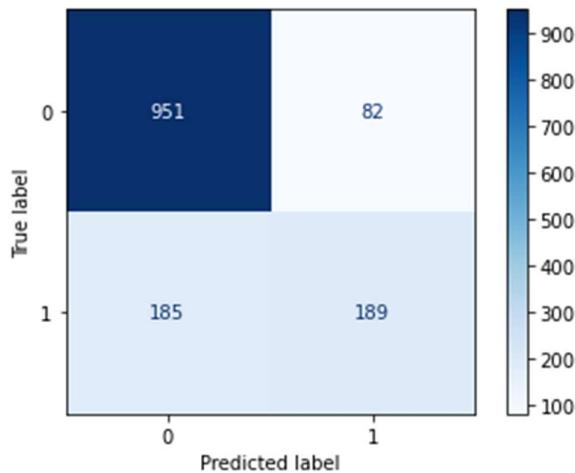


Figure : Matrice de confusion de SVM avec corrélation entre variables

5.6.2 Rapport de classification du SVM

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83715 | 0.92062 | 0.87690 | 1033 |
| 1 | 0.69742 | 0.50535 | 0.58605 | 374 |
| accuracy | | | 0.81023 | 1407 |
| macro avg | 0.76728 | 0.71298 | 0.73147 | 1407 |
| weighted avg | 0.80001 | 0.81023 | 0.79959 | 1407 |

Figure : Rapport de classification de SVM avec corrélation entre variables

5.7 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8086 | 0.7985 | 0.8647 | 0.6053 | 0.7985 | 0.8431 |
| DecisionTreeClassifier | 0.8216 | 0.7914 | 0.8618 | 0.5744 | 0.7914 | 0.826 |
| NaiveBayes(MultiNomial) | 0.7870 | 0.7985 | 0.8622 | 0.6255 | 0.7985 | 0.8425 |
| LogisticRegression | 0.8108 | 0.8090 | 0.8728 | 0.6165 | 0.8090 | 0.8599 |
| RandomForestClassifier | 0.8567 | 0.8142 | 0.8788 | 0.6016 | 0.8142 | 0.8597 |
| XGBClassifier | 0.8200 | 0.8144 | 0.8776 | 0.6167 | 0.8145 | 0.8683 |
| SVM | 0.8186 | 0.8102 | 0.8769 | 0.5860 | 0.8102 | False |

Correlation between features (with 23 features)

Figure : Table résumé de tous les modèles avec corrélation entre variables

→ Le modèle RandomForestClassifier donne les meilleurs résultats si on utilise la corrélation entre les variables pour la sélection des variables (avec 23 variables restantes)

5 Application des modeles en utilisant la corrélation avec la variable cible + entre les variables

5.1 Modele KNeighbors Classifier (KNN)

```
: from sklearn.neighbors import KNeighborsClassifier
knn_model=KNeighborsClassifier(n_neighbors=49,metric='manhattan')
knn_model.fit(x_train,y_train)
print('score du train ',knn_model.score(x_train,y_train))
print('score du test' ,knn_model.score(x_test,y_test))
score du train  0.8051605038602194
score du test 0.8004739336492891
```

Figure : Score de KNN avec corrélation entre les variables et variable cible

5.1.1 Matrice de confusion du KNN

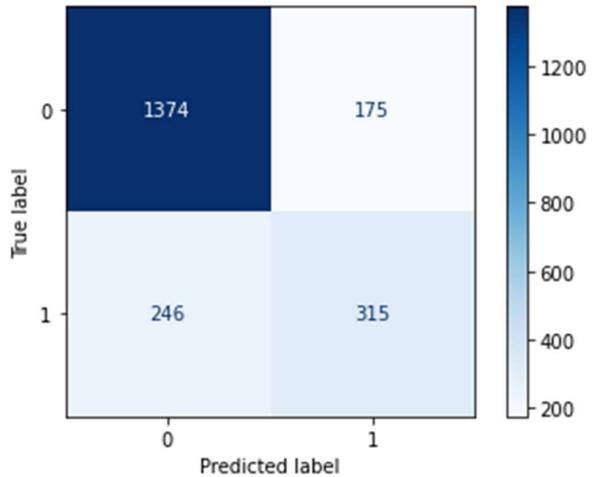


Figure : Matrice de confusion de kNN avec corrélation entre variables et variable cible

5.1.2 Rapport de classification du KNN

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|----------|
| 0 | 0.848148 | 0.887024 | 0.867151 | 1549 |
| 1 | 0.642857 | 0.561497 | 0.599429 | 561 |
| accuracy | | 0.800474 | | 2110 |
| macro avg | | 0.745503 | 0.724261 | 0.733290 |
| weighted avg | | 0.793566 | 0.800474 | 0.795970 |

Figure : Rapport de classification de kNN avec corrélation entre variables et variable cible

5.1.3 Courbe Roc du KNN

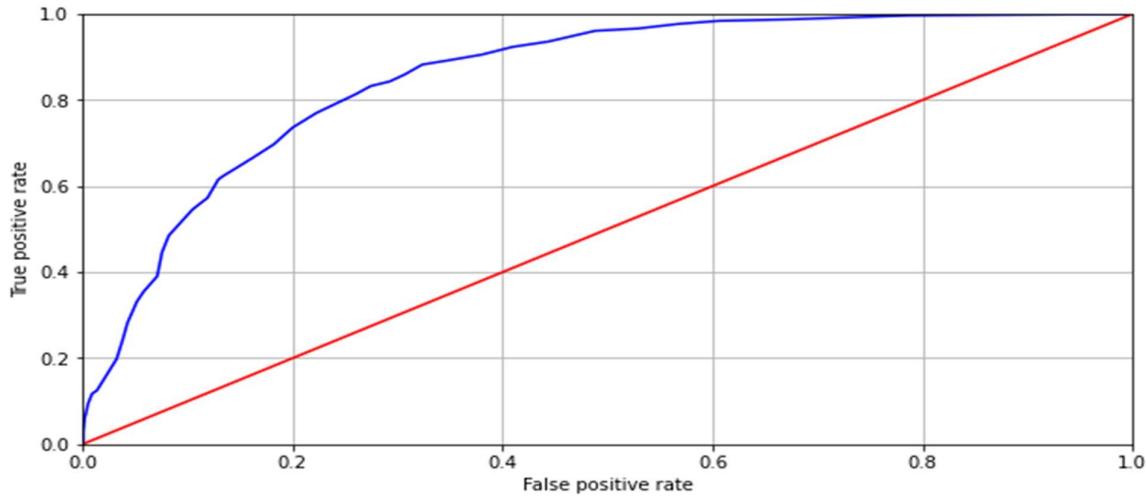


Figure : Courbe ROC de kNN avec corrélation entre variables et variable cible

5.2 Modele DecisionTreeClassifier(DT)

```

: from sklearn.tree import DecisionTreeClassifier

tree_model=DecisionTreeClassifier(criterion='gini',max_depth=6,random_state=0)
tree_model.fit(x_train,y_train)
print('train score : ', tree_model.score(x_train,y_train))
print('test score : ', tree_model.score(x_test,y_test))

train score :  0.8082080455099553
test score :  0.79478672985782

```

Figure : Score de DT avec corrélation entre les variables et variable cible

5.2.1 Matrice de confusion de DT

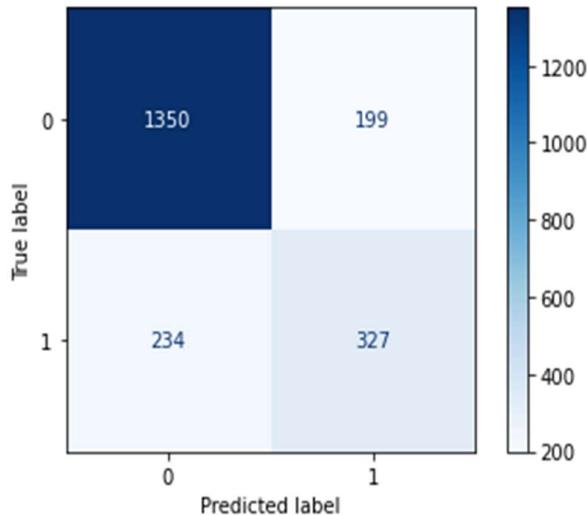


Figure : Matrice de confusion de DT avec corrélation entre variables et variable cible

5.2.2 Rapport de classification du DT

| | precision | recall | f1-score | support |
|--------------|-----------|----------|----------|----------|
| 0 | 0.852273 | 0.871530 | 0.861794 | 1549 |
| 1 | 0.621673 | 0.582888 | 0.601656 | 561 |
| accuracy | | | | 0.794787 |
| macro avg | 0.736973 | 0.727209 | 0.731725 | 2110 |
| weighted avg | 0.790962 | 0.794787 | 0.792629 | 2110 |

Figure : Rapport de classification de DT avec corrélation entre variables et variable cible

5.2.3 Courbe Roc du DT

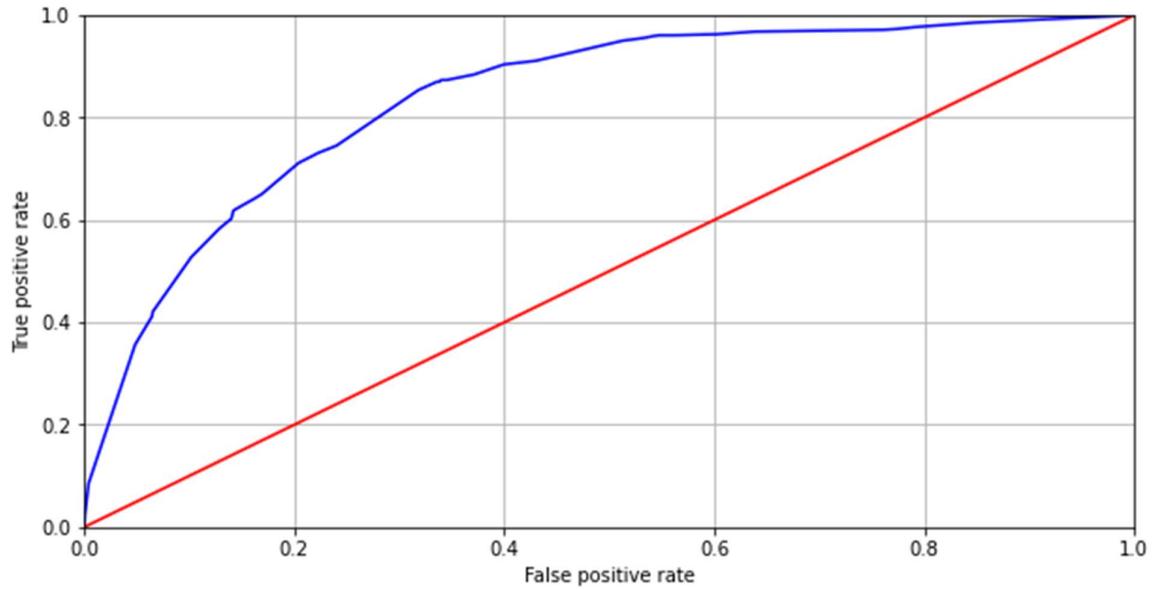


Figure Courbe ROC de DT avec corrélation entre variables et variable cible

5.2.4 Arbre de Decision du DT

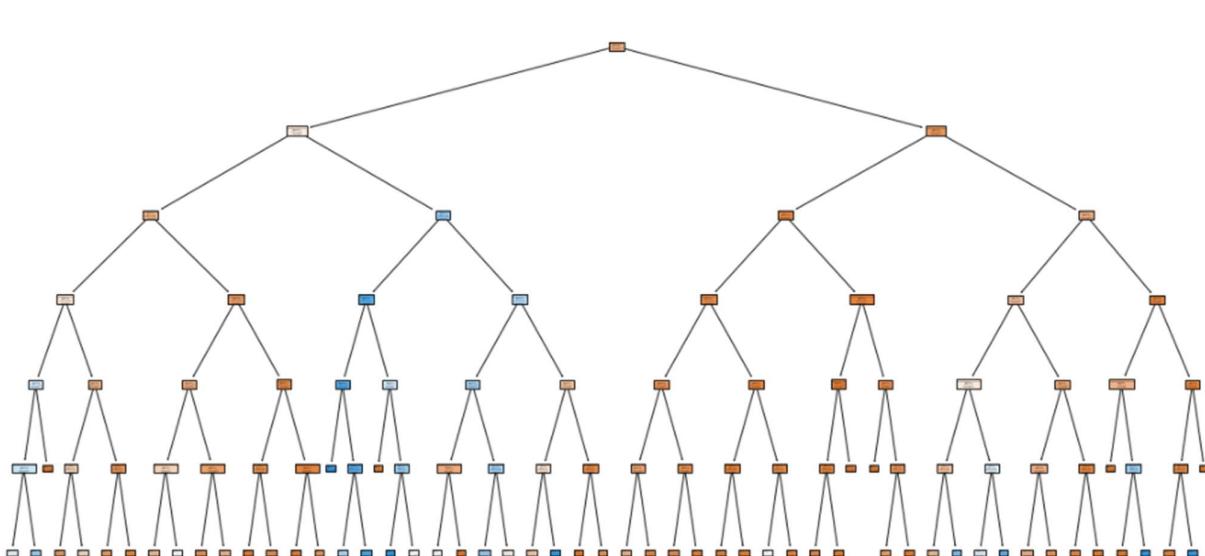


Figure DT avec corrélation entre variables et variable cible

5.2.5 Text d'export du DT

```

--- tenure months <= 0.22
    --- Internet Service_Fiber optic <= 0.50
        --- Tenure Months <= 0.04
            --- Internet Service_No <= 0.50
                --- Total Charges <= 0.02
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 1
                    --- Payment Method_Electronic check > 0.50
                        --- class: 1
                --- Total Charges > 0.02
                    --- class: 0
            --- Internet Service_No > 0.50
                --- Total Charges <= 0.00
                    --- Total Charges <= 0.00
                        --- class: 0
                    --- Total Charges > 0.00
                        --- class: 0
                --- Total Charges > 0.00
                    --- Total Charges <= 0.00
                        --- class: 0
                    --- Total Charges > 0.00
                        --- class: 0
        --- Tenure Months > 0.04
            --- Internet Service_No <= 0.50
                --- Total Charges <= 0.04
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 0
                    --- Payment Method_Electronic check > 0.50
                        --- class: 0
                --- Total Charges > 0.04
                    --- Payment Method_Electronic check <= 0.50
                        --- class: 0
                    --- Payment Method_Electronic check > 0.50
                        --- class: 0
            --- Internet Service_No > 0.50
                --- Dependents_Yes <= 0.50
--- Total Charges <= 0.01
    --- Dependents_Yes <= 0.50
        --- Total Charges <= 0.01
            --- class: 1
        --- Total Charges > 0.01
            --- Total Charges <= 0.01
                --- class: 1
            --- Total Charges > 0.01
                --- class: 1
    --- Dependents_Yes > 0.50
        --- Total Charges <= 0.01
            --- class: 0
        --- Total Charges > 0.01
            --- Total Charges <= 0.01
                --- class: 1
            --- Total Charges > 0.01
                --- class: 0
        --- Total Charges > 0.01
            --- Dependents_Yes <= 0.50
                --- Total Charges <= 0.01
                    --- class: 0
                --- Payment Method_Electronic check <= 0.50
                    --- class: 0
                    --- Payment Method_Electronic check > 0.50
                    --- class: 0
            --- Total Charges > 0.01
                --- Tenure Months <= 0.20
                    --- class: 1
                --- Tenure Months > 0.20
                    --- class: 0
            --- Dependents_Yes > 0.50
                --- Total Charges <= 0.13
                    --- Total Charges <= 0.12
                        --- class: 0
                    --- Total Charges > 0.12
                        --- class: 1

```

Figure Text export de DT avec corrélation entre variables et variable cible

5.3 Modele NaiveBayes (NB)

```

]: from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
      'bernoulli': BernoulliNB(),
      'multinomial': MultinomialNB()}
scores = {}
for key, model in nb.items(): #nb.items; parcourir cle et valeur
    s = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)
scores

]: {'gaussian': 0.711895299409847,
 'bernoulli': 0.7830089554702654,
 'multinomial': 0.7775240394535924}

]: bayes_model = BernoulliNB()
bayes_model.fit(x_train, y_train)
print('train score : ', bayes_model.score(x_train,y_train))
print('test score : ', bayes_model.score(x_test,y_test))

train score :  0.7830150345388054
test score :  0.781042654028436

```

Figure : Score de NB avec corrélation entre les variables et variable cible

5.3.1 Matrice de confusion du NB

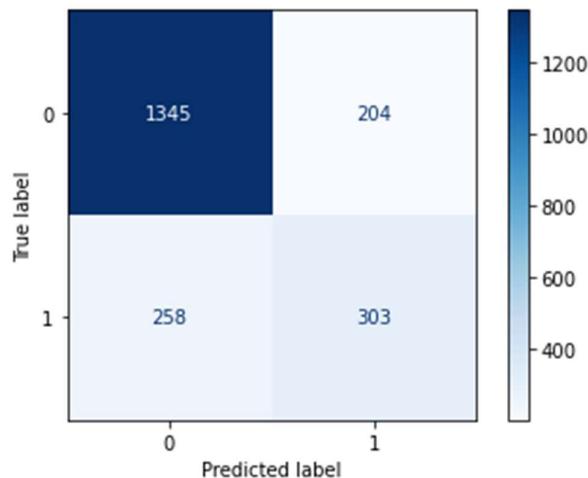


Figure : Matrice de confusion de NB avec corrélation entre variables et variable cible

5.3.2 Rapport de classification du NB

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83905 | 0.86830 | 0.85343 | 1549 |
| 1 | 0.59763 | 0.54011 | 0.56742 | 561 |
| accuracy | | | 0.78104 | 2110 |
| macro avg | 0.71834 | 0.70420 | 0.71042 | 2110 |
| weighted avg | 0.77486 | 0.78104 | 0.77738 | 2110 |

Figure Rapport de classification de NB avec corrélation entre variables et variable cible

5.3.3 Courbe Roc du NB

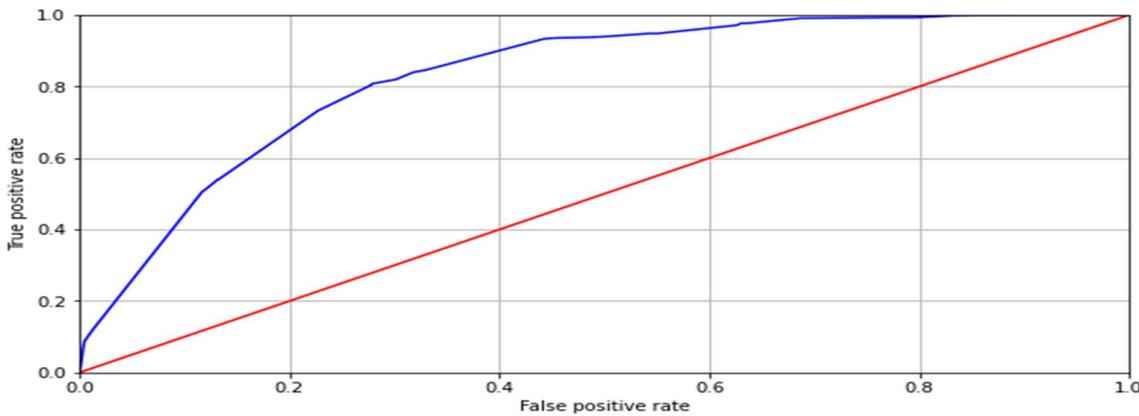


Figure : Courbe de ROC de NB avec corrélation entre variables et variable cible

5.4 Modele LogisticRegression (LR)

```
] from sklearn.linear_model import LogisticRegression  
  
logre_model = LogisticRegression(random_state=0, C=0.10101, penalty = 'l1', solver='saga')  
logre_model.fit(x_train, y_train)  
  
print('train score : ' , logre_model.score(x_train, y_train) )  
print('test score : ' , logre_model.score(x_test,y_test) )  
  
train score :  0.8015034538805363  
test score :  0.8052132701421801
```

Figure : Score de LR avec corrélation entre les variables et variable cible

5.4.1 Matrice de confusion du LR

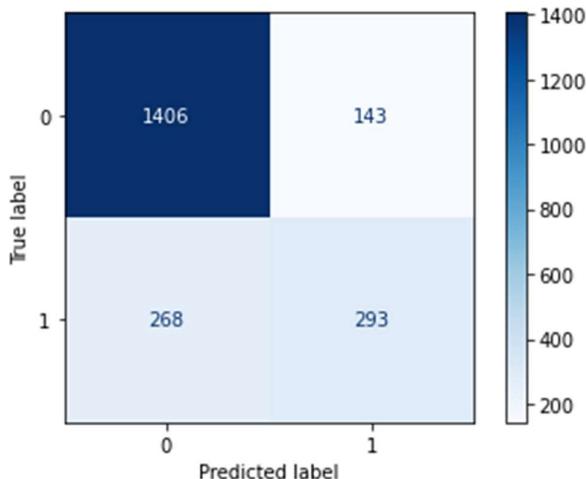


Figure : Matrice de confusion de LR avec corrélation entre variables et variable cible

5.4.2 Rapport de classification du LR

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.83990 | 0.90768 | 0.87248 | 1549 |
| 1 | 0.67202 | 0.52228 | 0.58776 | 561 |
| accuracy | | | 0.80521 | 2110 |
| macro avg | 0.75596 | 0.71498 | 0.73012 | 2110 |
| weighted avg | 0.79527 | 0.80521 | 0.79678 | 2110 |

Figure : Rapport de classification de LR avec corrélation entre variables et variable cible

5.4.3 Courbe Roc du LR

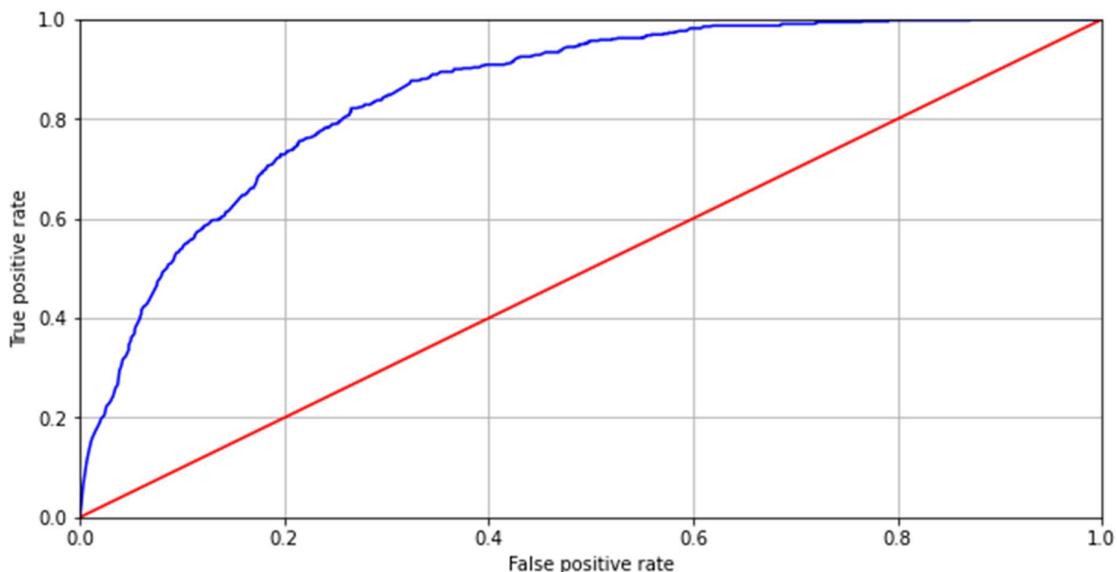


Figure : Courbe ROC de LR avec corrélation entre variables et variable cible

5.5 Modele RandomForestClassifier (RFC)

```
: from sklearn.ensemble import RandomForestClassifier  
  
random_model=RandomForestClassifier(criterion='entropy', max_depth=6,random_state=90)  
random_model.fit(x_train,y_train)  
  
print('train score : ',random_model.score(x_train,y_train))  
print('test score : ',random_model.score(x_test,y_test))  
  
train score :  0.8074666666666667  
test score :  0.8045486851457001
```

Figure : Score de RFC avec corrélation entre les variables et variable cible

5.5.1 Matrice de confusion du RFC

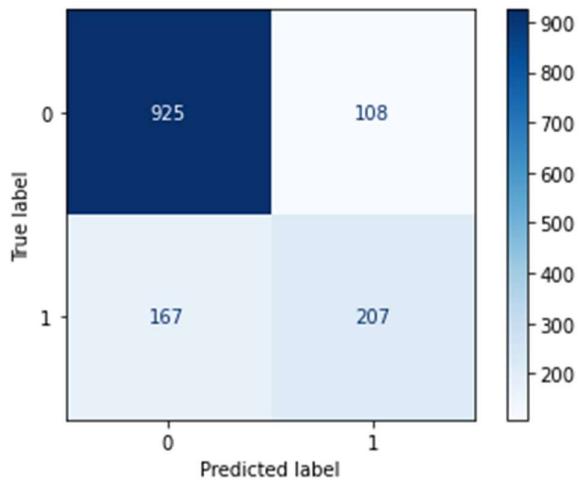


Figure : Matrice de confusion de RFC avec corrélation entre variables et variable cible

5.5.2 Rapport de classification du RFC

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84707 | 0.89545 | 0.87059 | 1033 |
| 1 | 0.65714 | 0.55348 | 0.60087 | 374 |
| accuracy | | | 0.80455 | 1407 |
| macro avg | 0.75211 | 0.72446 | 0.73573 | 1407 |
| weighted avg | 0.79658 | 0.80455 | 0.79889 | 1407 |

Figure : Rapport de RFC avec corrélation entre variables et variable cible

5.5.3 Courbe Roc du RFC

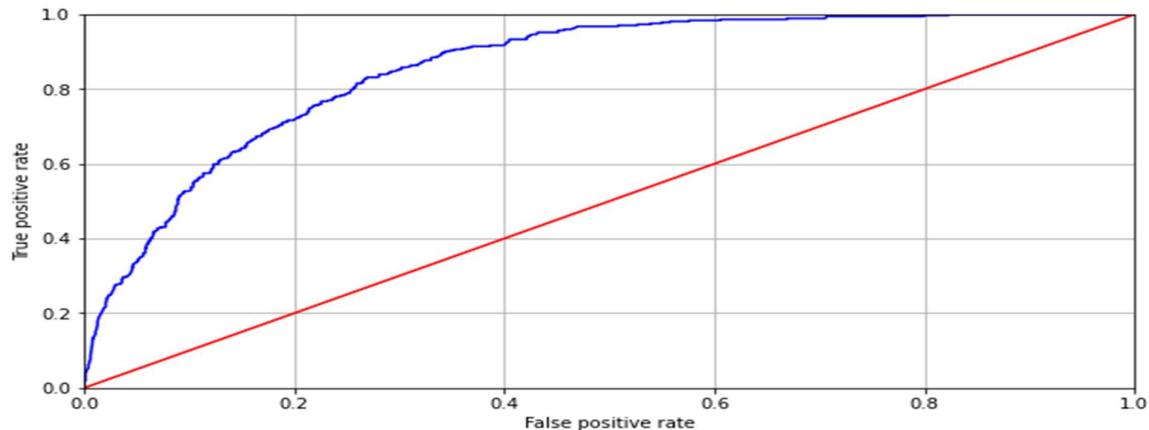


Figure : Courbe ROC de RFC avec corrélation entre variables et variable cible

5.6 Modele XGboost(xgb)

```

: from xgboost import XGBClassifier

xgb_model=XGBClassifier(max_depth=2,n_estimators=100,learning_rate=0.1)
xgb_model.fit(x_train,y_train)
print(xgb_model.score(x_train,y_train))
print(xgb_model.score(x_test,y_test))

[00:25:56] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner
valuation metric used with the objective 'binary:logistic' was changed from 'error' to 'logl
restore the old behavior.
0.8058666666666666
0.8024164889836531

```

Figure : Score de XGB avec corrélation entre les variables et variable cible

5.6.1 Matrice de confusion du xgb

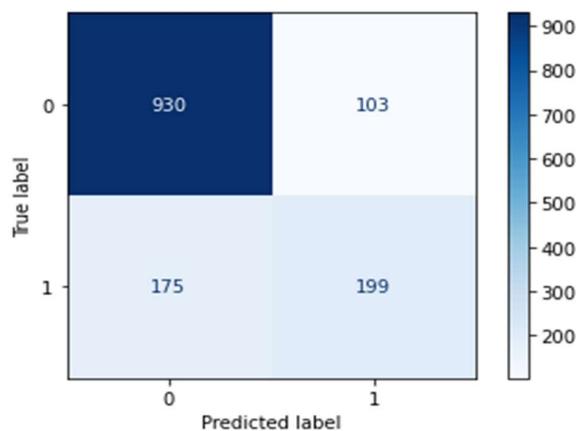


Figure : Matrice de confusion de XGB avec corrélation entre variables et variable cible

5.6.2 Rapport de classification du xgb

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84163 | 0.90029 | 0.86997 | 1033 |
| 1 | 0.65894 | 0.53209 | 0.58876 | 374 |
| accuracy | | | 0.80242 | 1407 |
| macro avg | 0.75028 | 0.71619 | 0.72936 | 1407 |
| weighted avg | 0.79307 | 0.80242 | 0.79522 | 1407 |

Figure : Rapport de classification de XGB avec corrélation entre variables et variable cible

5.6.3 Courbe Roc du xgb

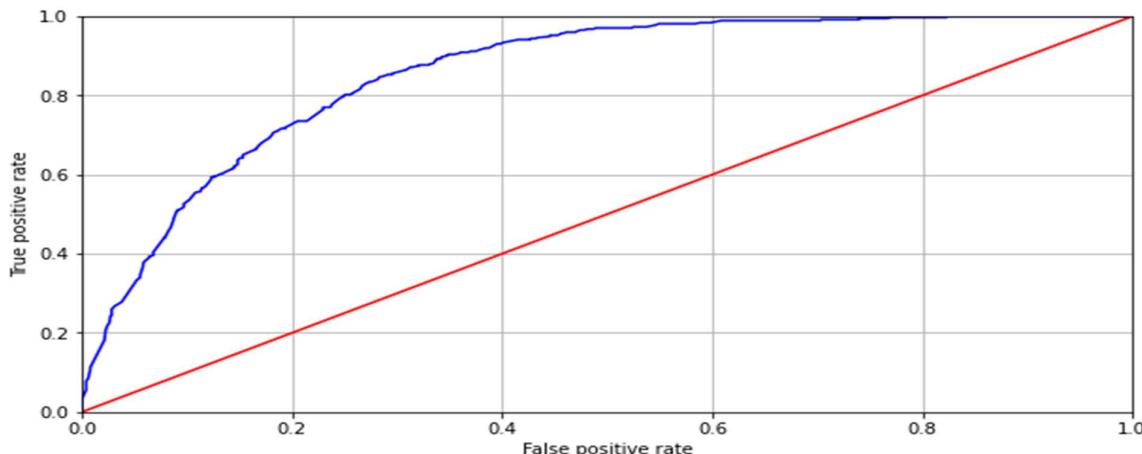


Figure : Courbe ROC de XGB avec corrélation entre variables et variable cible

5.7 Modele Support Vector Machine (SVM)

```
[]: from sklearn import svm
svm_model=svm.SVC(C=100,kernel='rbf',gamma=1)
svm_model.fit(x_train,y_train)
print(svm_model.score(x_train,y_train))
print(svm_model.score(x_test,y_test))

0.8035555555555556
0.8045486851457001
```

Figure : Score de SVM avec corrélation entre les variables et variable cible

5.7.1 Matrice de confusion du SVM

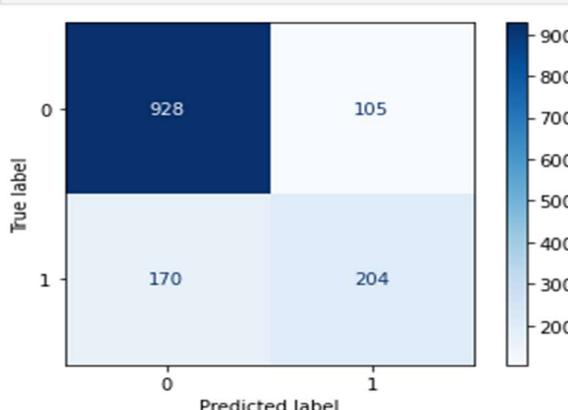


Figure : Matrice de confusion de SVM avec corrélation entre variables et variable cible

5.7.2 Rapport de classification du SVM

| | precision | recall | f1-score | support |
|--------------|-----------|---------|----------|---------|
| 0 | 0.84517 | 0.89835 | 0.87095 | 1033 |
| 1 | 0.66019 | 0.54545 | 0.59736 | 374 |
| accuracy | | | 0.80455 | 1407 |
| macro avg | 0.75268 | 0.72190 | 0.73416 | 1407 |
| weighted avg | 0.79600 | 0.80455 | 0.79823 | 1407 |

Figure : Rapport de classification du SVM avec corrélation entre variables et variable cible

5.8 Table de résumé :

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8051 | 0.8004 | 0.8671 | 0.5964 | 0.8004 | 0.8527 |
| DecisionTreeClassifier | 0.8082 | 0.7947 | 0.8617 | 0.6016 | 0.7947 | 0.8381 |
| NaiveBayes(MultiNomial) | 0.7830 | 0.7810 | 0.8534 | 0.5674 | 0.7810 | 0.8301 |
| LogisticRegression | 0.8015 | 0.8052 | 0.8724 | 0.5877 | 0.8052 | 0.8524 |
| RandomForestClassifier | 0.8074 | 0.8045 | 0.8705 | 0.6008 | 0.8045 | 0.8569 |
| XGBClassifier | 0.8058 | 0.8024 | 0.8699 | 0.5887 | 0.8024 | 0.8561 |
| SVM | 0.8035 | 0.8045 | 0.8009 | 0.5937 | 0.8045 | False |

Correlation between features and target and

Correlation features betwen them (with 7 features)

Figure : Table des modèles avec corrélation entre variables et variable cible

- Le modèle **RandomForestClassifier** donne les meilleurs résultats si on utilise la corrélation avec la variable cible + la corrélation entre les variables pour la sélection des variables (avec 7 variables restantes)

V. Evaluation:

Dans cette partie on va interpréter les résultats obtenus dans la partie de modélisation et prendre des décisions pour valider le meilleur modèle , Le meilleur **score du train** obtenu est autour de **85 %** Et pour **le test score** autour de **81 %** en utilisant le modèle **RandomForestClassifier** mais en gardant **26 variables** en appliquant **la variance** de la sélection des variables ce qui gêne dans la pratique .Cela est dû parce que notre base est déséquilibré .

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8086 | 0.7985 | 0.8647 | 0.6053 | 0.7985 | 0.8431 |
| DecisionTreeClassifier | 0.8216 | 0.7914 | 0.8618 | 0.5744 | 0.7914 | 0.826 |
| NaiveBayes(MultiNomial) | 0.7870 | 0.7985 | 0.8622 | 0.6255 | 0.7985 | 0.8425 |
| LogisticRegression | 0.8108 | 0.8090 | 0.8728 | 0.6165 | 0.8090 | 0.8599 |
| RandomForestClassifier | 0.8567 | 0.8142 | 0.8788 | 0.6016 | 0.8142 | 0.8597 |
| XGBClassifier | 0.8200 | 0.8144 | 0.8776 | 0.6167 | 0.8145 | 0.8683 |
| SVM | 0.8186 | 0.8102 | 0.8769 | 0.5860 | 0.8102 | False |

Correlation between features (with 23 features)

Figure : Table des modèles avec la variance

Mais si le client s'intéresse à un nombre limité des variables, nous sommes arrivés à un modèle qui utilise seulement **7 variables** pour la prédiction en appliquant **2 techniques de sélection des** 84

variable en même temps : la corrélation avec la variable cible ensuite la corrélation entre les variables eux même, c'est le RandomForestClassifier aussi avec au train score de 80% et un test score de 80 % .

| | train_score | test_score | f1_score_rester | f1_score_quitter | accuary | AUC |
|-------------------------|-------------|------------|-----------------|------------------|---------|--------|
| KNeighborsClassifier | 0.8051 | 0.8004 | 0.8671 | 0.5964 | 0.8004 | 0.8527 |
| DecisionTreeClassifier | 0.8082 | 0.7947 | 0.8617 | 0.6016 | 0.7947 | 0.8381 |
| NaiveBayes(MultiNomial) | 0.7830 | 0.7810 | 0.8534 | 0.5674 | 0.7810 | 0.8301 |
| LogisticRegression | 0.8015 | 0.8052 | 0.8724 | 0.5877 | 0.8052 | 0.8524 |
| RandomForestClassifier | 0.8074 | 0.8045 | 0.8705 | 0.6008 | 0.8045 | 0.8569 |
| XGBClassifier | 0.8058 | 0.8024 | 0.8699 | 0.5887 | 0.8024 | 0.8561 |
| SVM | 0.8035 | 0.8045 | 0.8009 | 0.5937 | 0.8045 | False |

Correlation between features and target and

Correlation features betwen them (with 7 features)

Figure : Table des modèles avec corrélation entre variables et variable cible + corrélation entre variables

➔ Ici le choix revient au client

VI. Déploiement :

L'étape du déploiement est très importante, notre objectif est de fournir une interface **simple** et efficace qui relie l'utilisateur et le modèle finale choisi



On a décidé de choisir le modelé qui correspond à **7 entrées** et on va utiliser le Framework **Django** afin de créer une application web qui permet d'estimer si un client va quitter l'entreprise.

Avant de commencer le déploiement il faut exporter le modèle choisi grâce à la bibliothèque Pickle

```
In [301]: import pickle  
  
# Enregistrer le modèle dans un fichier  
filename = 'notreModeleFinale.sav'  
pickle.dump(finalModel, open(filename, 'wb'))
```

Figure : Exportation du modèle choisi avec pickle

Maintenant, notre modèle entraîné est enregistré dans un fichier binaire et on peut le charger ultérieurement dans le projet de déploiement

1. Configuration d'un environnement virtuel

Pour créer un environnement virtuel pour notre projet, il faut ouvrir une nouvelle invite de commande et taper

```
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\BAYCII>py -m venv churn-PredictionApp
```

Figure : Création de notre environnement virtuel

2. Installation de Django

Django peut être installé facilement en utilisant **pip** dans notre environnement virtuel.

Dans l'invite de commande, il faut activer l'environnement virtuel exécutez les commandes suivantes :

```
C:\Users\BAYCII>cd churn-PredictionApp

C:\Users\BAYCII\churn-PredictionApp>cd Scripts

C:\Users\BAYCII\churn-PredictionApp\Scripts>activate

(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts>
```

Figure : activation de notre environnement virtuel

```
(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts>py -m pip install Django
Collecting Django
  Downloading Django-3.1.5-py3-none-any.whl (7.8 MB)
    |████████████████████████████████| 7.8 MB 70 kB/s
Collecting pytz
  Downloading pytz-2020.5-py2.py3-none-any.whl (510 kB)
    |████████████████████████████████| 510 kB 726 kB/s
Collecting asgiref<4,>=3.2.10
  Downloading asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.1-py3-none-any.whl (42 kB)
    |████████████████████████████████| 42 kB 1.6 MB/s
Installing collected packages: pytz, asgiref, sqlparse, Django
Successfully installed Django-3.1.5 asgiref-3.3.1 pytz-2020.5 sqlparse-0.4.1
```

Figure : installation de Django dans notre environnement virtuel grâce à pip

3. Création et configuration d'un projet Django

Pour créer un projet Django, il faut lancer la commande suivante :

```
(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts>django-admin startproject ChurnAPP
```

Figure : Crédit d'un projet Django

Pour créer une application il faut aller dans notre projet et lancer la commande suivante :

```
(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts>cd churnAPP  
(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts\ChurnAPP>python manage.py startapp predict  
(churn-PredictionApp) C:\Users\BAYCII\churn-PredictionApp\Scripts\ChurnAPP>
```

Figure : Lancement d'une nouvelle application dans Django

Voici le contenu de notre application Django :

| Name | Date modified | Type | Size |
|-------------|----------------|-------------|------|
| migrations | 1/9/2021 18:50 | File folder | |
| __init__.py | 1/9/2021 18:50 | Python File | 0 KB |
| admin.py | 1/9/2021 18:50 | Python File | 1 KB |
| apps.py | 1/9/2021 18:50 | Python File | 1 KB |
| models.py | 1/9/2021 18:50 | Python File | 1 KB |
| tests.py | 1/9/2021 18:50 | Python File | 1 KB |
| views.py | 1/9/2021 18:50 | Python File | 1 KB |

Figure : Contenu de l'application Django

| Name | Status | Date modified | Type | Size |
|-------------|--------|------------------|-------------|------|
| __pycache__ | ✓ | 1/7/2021 19:43 | File folder | |
| __init__.py | ✓ | 12/31/2020 12:47 | Python File | 0 KB |
| asgi.py | ✓ | 12/31/2020 12:47 | Python File | 1 KB |
| settings.py | ✓ | 12/31/2020 13:31 | Python File | 4 KB |
| urls.py | ✓ | 1/7/2021 19:43 | Python File | 2 KB |
| wsgi.py | ✓ | 12/31/2020 12:47 | Python File | 1 KB |

Figure : Contenu de projet Django

4. Implementation du code de notre application

Maintenant on est prêt pour coder notre application, on a créé une interface index (notre front end) et un micro service predict qui prend en paramètre les entrée et se charge de répondre par le résultat du modèle , du coup il faut d'abord configurer le fichier urls.py

```
from django.contrib import admin
from django.urls import path
from myChurnApp.views import index
from myChurnApp.views import predict

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index),
    path('predict/<int:Dependents_Yes>/<int:Internet_Service>/<int:Contract_Two_year>/<int:Payment_Method_Electronic_check>/<int:Tenure_Months>', predict)
]
```

Figure : Contenu du fichier urls.py

On a utilisé jQuery pour créer une seule page web dynamique qui échange les données avec le micro-service predict

```
335     proximity: 20, // How close two dots need to be before they join
336     parallaxMultiplier: 20, // Lower the number is more extreme parallax
337     particleRadius: 2 // Dot size
338   );
339   //## sourceURL=pen.js
340   </script>
341   <script>
342     function load() {
343       var currentHeader =$('#header').html();
344       $('#header').html('<div class="vertical-centered-box"><div class="content"><div class="loader-circle"></div><div class="loading">Please wait...</div></div></div>');
345       location.hash = '';
346       setTimeout(function(){
347         $.get( "/predict/"+$( '#d' ).val () +"/"+$( '#is' ).val () +"/"+$( '#c' ).val () +"/"+$( '#pm' ).val () +"/"+$( '#tm' ).val () +"/"+$( '#tt' ).val () );
348       }, 2000);
349       setTimeout(function(){
350         $('#header').html(currentHeader);
351       },2000);
352     }, 2000);
353
354
355
356
357 } </script>
```

Figure : Contenu de l'interface index.html

Le micro service predict prends les paramètres passés dans l'url lors de son appel, et il charge le modèle avec pickle puis il retourne la réponse du modèle :

```
1 from django.shortcuts import render
2 import pandas as pd
3 import numpy as np
4 import pickle
5
6
7 # Create your views here.
8 def index(request):
9     nomGroupe="Groupe de BAYCII"
10    return render(request,'myChurnApp/index.html',{'nomGroupe':nomGroupe})
11
12 def predict(request,Dependents_Yes,Internet_Service,Contract_Two_year,Payment_Method_Electronic_check,Tenure_Months,Total_Charges):
13
14     print("%d %d %d %d %f %f" % (Dependents_Yes,Internet_Service,Contract_Two_year,Payment_Method_Electronic_check,float(Tenure_Months),float(Total_Charges)))
15     internetServiceFibre = 0
16     internetServiceNo = 0
17     if Internet_Service == 1:
18         internetServiceFibre = 1
19     if Internet_Service == 2:
20         internetServiceNo = 1
21     tMonth = ( ( float(Tenure_Months) - float(1) ) / ( float(72) - float(1) ) )
22     totalC = ( ( float(Total_Charges) - float(18.8) ) / ( float(8672.45) - float(18.8) ) )
23     RandForestModel=pickle.load(open('C:/Users/BAYCII/OneDrive/Documents/Projet DS/NoteBooks/random_forest.sav','rb'))
24     churnValue=RandForestModel.predict([[Dependents_Yes,internetServiceFibre,internetServiceNo,Contract_Two_year,Payment_Method_Electronic_check,Tenure_Months,Total_Charges]])
25
26
```

Figure : Contenu de views.py

Maintenant, tout est prêt, on va lancer notre serveur d'application (avec le port http 80) :

```
(myApp) C:\Users\BAYCII\OneDrive\Documents\Projet DS\App\churnApp>python manage.py runserver 80
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
January 09, 2021 - 19:12:17
Django version 3.1.4, using settings 'churnApp.settings'
Starting development server at http://127.0.0.1:80/
Quit the server with CTRL-BREAK.
```

Figure : Commande pour lancer le serveur avec le port 80

5. Test de notre application

On lance notre navigateur web et on visite l'url localhost (ou bien 127.0.0.1) et voici les scénarios d'utilisation de notre application web

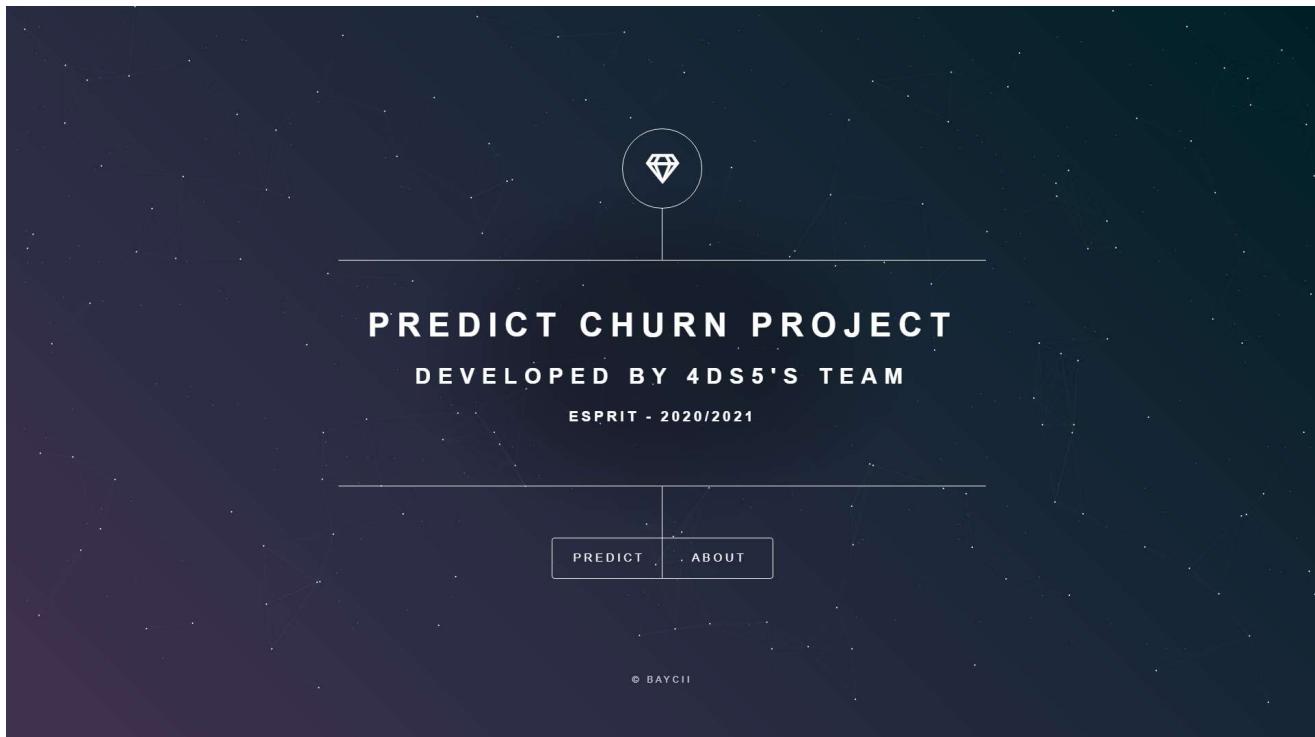


Figure : Page d'accueil de notre application

Lorsque on clique sur Predict un formulaire dynamique est affiché :

CUSTOMER'S INFORMATIONS

CONTRACT **PAYMENT METHOD**

One year Electronic check

INTERNET SERVICE **DEPENDENTS**

No Yes

TENURE MONTHS **TOTAL CHARGES**

3 128

PREDICT

Figure : Formulaire de notre application

Si on clique sur predict les inputs seront envoyé à notre micro-service predict avec AJAX

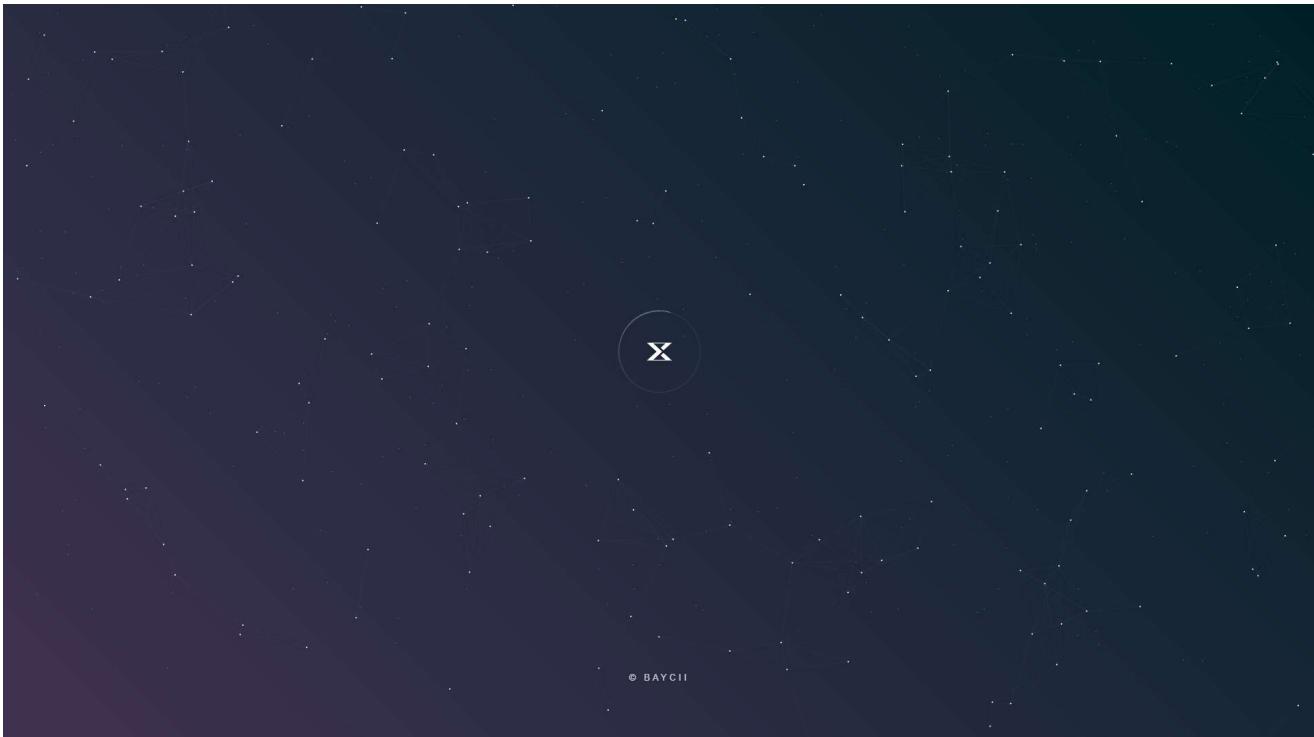


Figure : Chargement de la réponse

Les deux scenarios possibles selon la repense du modèle :

Scenario 1 Réponse Positive :

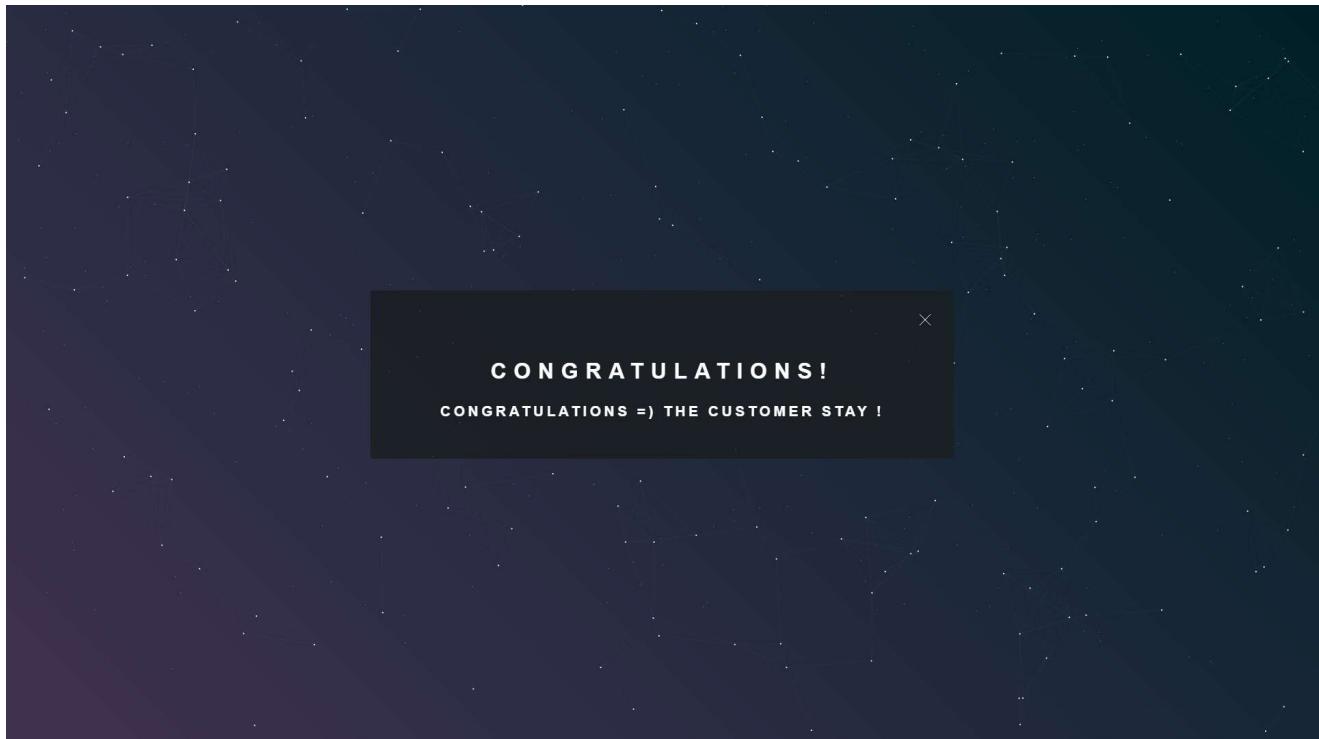


Figure : Retour du micro service (le client ne va pas quitter)

Scenario 2 Réponse négative :

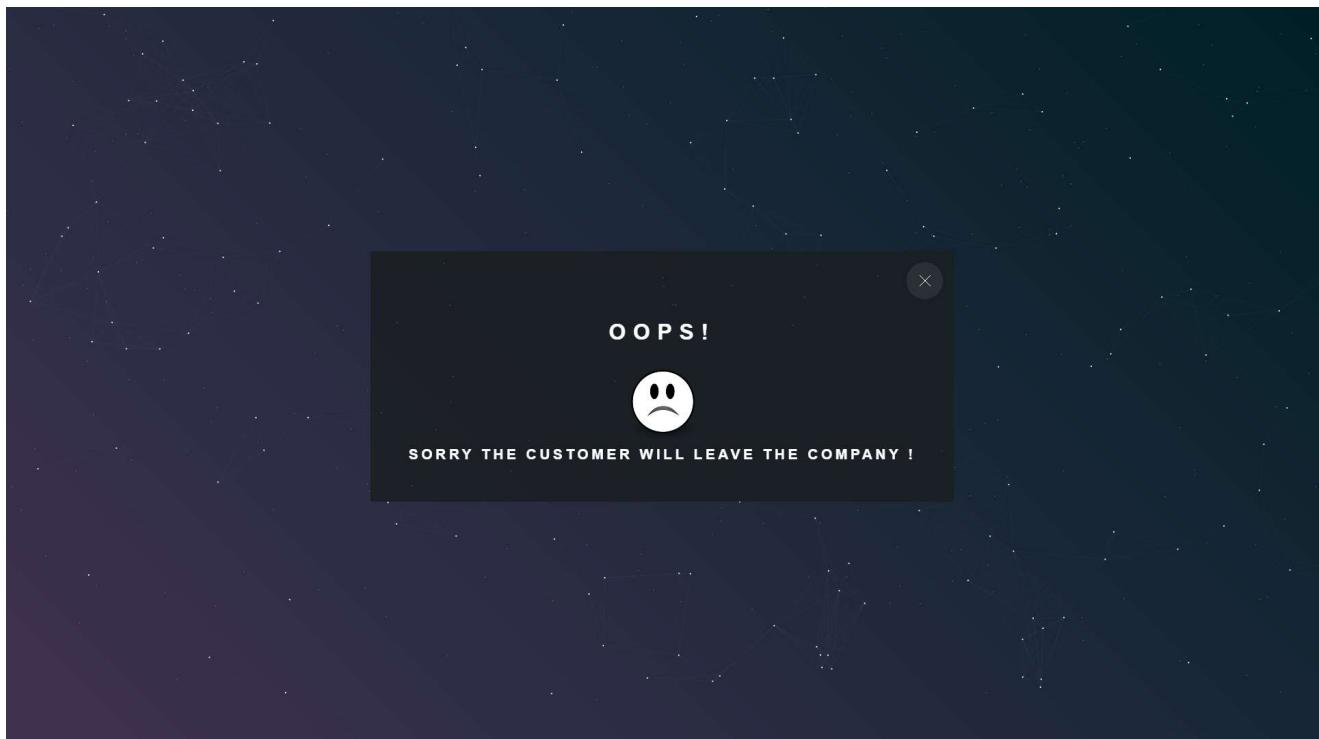


Figure : Retour du micro service (le client va quitter)

VII. Conclusion:

Le but de ce projet est de faire des analyses et proposer des directives pour pouvoir Prendre des décisions sur la réduction du taux de churn ou perte de client.

Aujourd’hui la perte de client est synonyme de perte de source de revenus. Pour cela nous avons utilisé des techniques de machine learning pour faire de la modélisation prédictive. Ces techniques permettent de déterminer à l'avance avec une certaine probabilité si le client est un potentiel churner ou restera actif.

Premièrement, nous avons pu voir les techniques déjà utilisé en machine learning pour la résolution du problème de churn. Puis nous avons pu voir toutes les étapes de transformations et préparations des données. Nous avons testé et comparé plusieurs algorithmes pour trouver le mieux approprié pour la prédiction de l'état du client. Après évaluation de la performance sur chacun des modèles, nous avons sélectionné le modèle avec une bonne précision prédictive et à la fois facilement interprétable. Enfin nous avons montré la procédure de déploiement de ce modèle choisi.

Le modèle de classification construit a permis de distinguer les clients Churners des Actifs à partir de ces activités pendant le temps où ces clients était Actifs