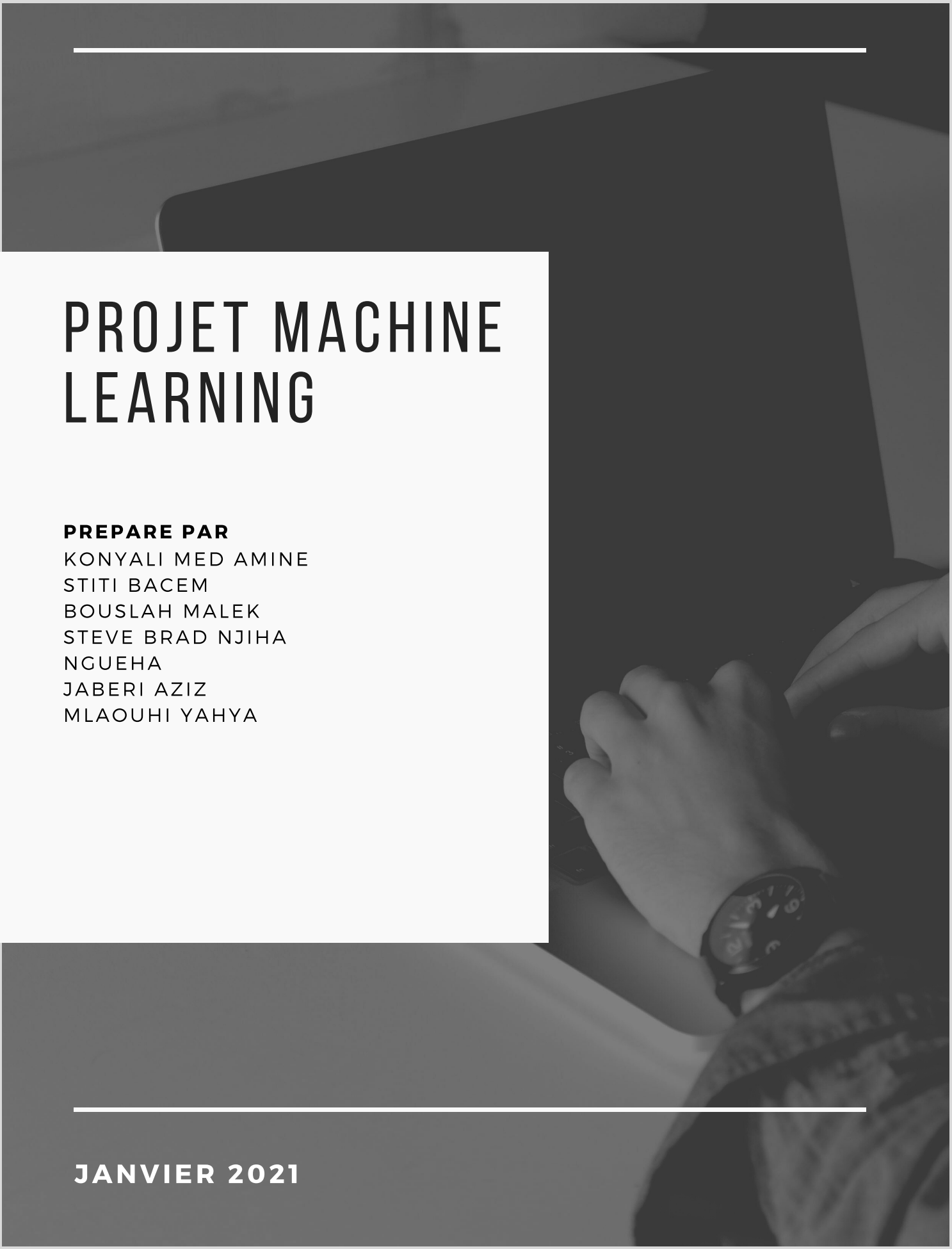
esprit
Se former autrement



PROJET MACHINE LEARNING

PREPARE PAR

KONYALI MED AMINE
STITI BACEM
BOUSLAH MALEK
STEVE BRAD NJIHA NGUEHA
JABERI AZIZ
MLAOUHI YAHYA

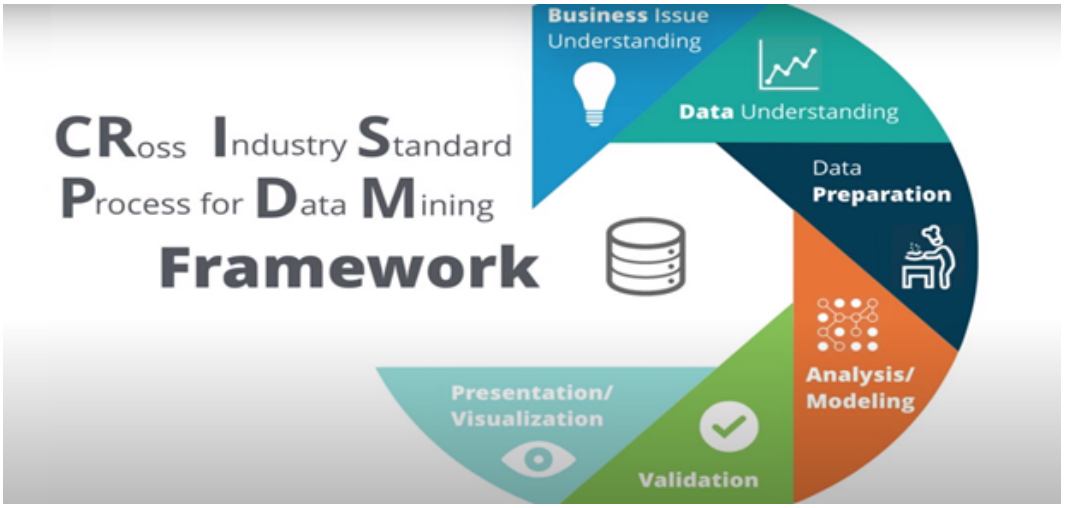
JANVIER 2021

TABLE DES MATIÈRES

INTRODUCTION

Les télécommunications mobiles sont devenues le moyen de communication dominant dans le monde, ce qui en fait un marché saturé où les entreprises sont confrontées à une concurrence commerciale féroce. C'est la raison pour laquelle les entreprises recherchent des services plus efficaces pour mettre un terme à la rotation des clients. Le taux de désabonnement des clients a un impact sur l'image de l'entreprise, il est donc préférable de l'empêcher pour éviter toute perte potentielle. Étant donné que l'obtention de nouveaux clients coûte environ 5 à 10 fois plus cher, la fidélisation de la clientèle est plus importante que l'acquisition.
Les entreprises cherchent donc toujours à améliorer leur service client et à améliorer la qualité de leurs produits.
Des études de recherche ont montré que le meilleur moyen d'éviter cela est d'étudier les données des clients, et c'est là qu'interviennent les méthodes statistiques classiques telles que K-NN, Naïve Bayes et bien d'autres méthodes.
L'utilisation de l'apprentissage automatique des données disponibles pour prédire le taux de désabonnement des clients est la première méthode pour conserver les anciens clients tout en en acquérant de nouveaux. L'exploration de données des abonnées permet aux ordinateurs de les utiliser pour prévoir les comportements, les résultats et les tendances futures.

MÉTHODOLOGIE CRISP



Nombreuses sont les entreprises qui engagent des projets de Datamining déploient plusieurs méthodes , l'une de celles est la methode CRISP .
CRISP-DM, qui signifie CRoss-Industry Standard Process for Data Mining, est une méthode mise à l'épreuve sur le terrain permettant d'orienter les travaux d'exploration de données.
Cette méthode se décompose en 6 étapes allant de la compréhension du problème métier au déploiement et la mise en production.

1-La compréhension du problème métier:
Qui permet de fixer les objectifs du projet en se posant les questions métiers.

2-La compréhension des données:
Qui consiste en la découverte des données à notre disposition ainsi qu'à leur exploration.

3-La préparation des données:
Qui est une phase de nettoyage et réorganisation des données, et si nécessaire de création de nouvelles valeurs calculées à partir d'autres données.

4-La modélisation:
Pour la sélection du modèle de fouille de données ainsi que ses paramètres.

5-L'évaluation:
Qui consiste à juger les performances du modèle et comparer les résultats produits avec les objectifs à atteindre.

6-Le déploiement à la fin du projet:
Phase pendant laquelle la solution est installée et de nouvelles pratiques sont mises en place suite aux conclusions tirées du projet.

COMPRÉHENSION DU PROBLÈME MÉTIER

Au cours de la dernière décennie, une croissance rapide de l'industrie des télécommunications a conduit à une plus grande base d'abonnés pour les fournisseurs de services et les téléphones portables sont en train de devenir le moyen de communication dominant dans le monde. En effet, les entreprises de télécommunications sont toujours en concurrence pour gagner des clients et maintiennent leur fidélité, car fidéliser l'ancien client coûte cinq à six fois moins cher que de trouver un nouveau client. Par conséquent, c'est nécessaire pour que la société de télécommunications ait la capacité de prédire le client qui lui sera fidèle.

COMPRÉHENSION DES DONNÉES

DONNEES INITIALES

L'ensemble de données utilisé dans cette étude est basé sur données historiques. Il s'agit d'un ensemble de données public de l'ensemble de données Kaggle (https://www.kaggle.com/blastchar/telco-customer-churn) au format CSV et spécifiquement déclaré comme données de désabonnement des clients des opérateurs de télécommunications.
L'ensemble de données se compose de 7043 lignes avec 33 attributs.

EXPLORATION DES DONÉES

Dans cette tâche, nous examinons les données de plus près. Pour chaque variable, nous regardons la plage de valeurs et leurs distributions, nous utiliserons une manipulation simple des données et des techniques statistiques de base pour des vérifications supplémentaires des données.

Notre ensemble de données a 33 attributs dont certains ne sont pas significatifs pour notre problème de désabonnement, ils doivent être nettoyés pour de meilleurs résultats.

```
Entrée [110]: data.isnull().sum()

Out[110]: CustomerID           0
          Count                0
          Country              0
          State                0
          City                 0
          ZipCode              0
          LatLong              0
          Latitude             0
          Longitude            0
          Gender               0
          SeniorCitizen        0
          Partner              0
          Dependents           0
          TenureMonths         0
          PhoneService         0
          MultipleLines        0
          InternetService      0
          OnlineSecurity       0
          OnlineBackup         0
          DeviceProtection     0
          TechSupport          0
          StreamingTV          0
          StreamingMovies      0
          Contract             0
          PaperlessBilling     0
          PaymentMethod        0
          MonthlyCharges       0
          TotalCharges         0
          ChurnLabel           0
          ChurnValue           0
          ChurnScore           0
          CLTV                 0
          ChurnReason       5174
          dtype: int64
```

.

Nous remarquons que seulement l'attribut churn reason a des valeurs manquantes.

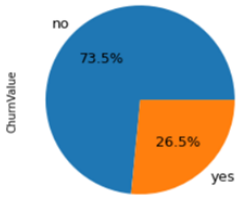


Figure 1 : Total des clients churn ou non

D'après la figure ci-dessus, nous avons remarqué que nos données sont déséquilibrées:
73,5% sont des clients non-churn et 26,5% sont des churn , donc un processus de sur ou sous-échantillonnage est nécessaire dans la phase de classification.

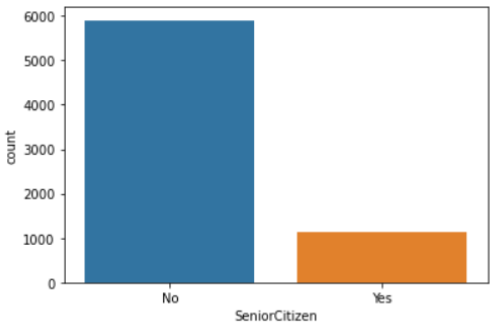


Figure 2 : Total des Senior Citizen

Nous remarquons que le nombre de personnes âgées est inférieur à celui des citoyens plus jeunes.

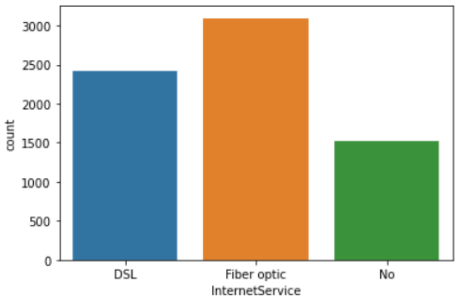


Figure 3 : Total des différents internet service

La fibre optique c'est le service internet le plus utilisé par les abonnés.

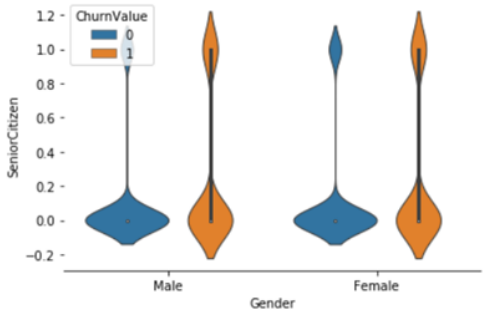


Figure4: Corrélation entre Senior Citizen et Gender

La figure montre que les jeunes, femmes et hommes, pourraient être plus nombreux à se tourner vers d'autres opérateurs que les personnes âgées.

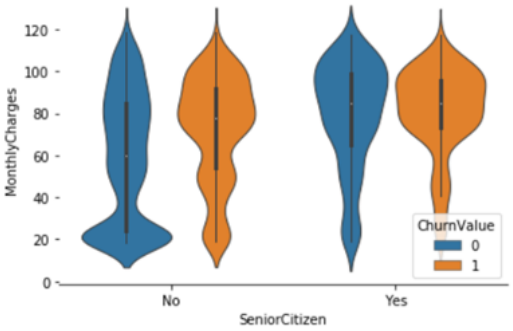


Figure 5 :Correlation entre Senior Citizen et Monthly Charges

La figure montre que les clients jeunes et vieux avec des frais mensuels élevés sont beaucoup plus susceptibles de se désabonner que les clients avec des frais mensuels totaux faibles.

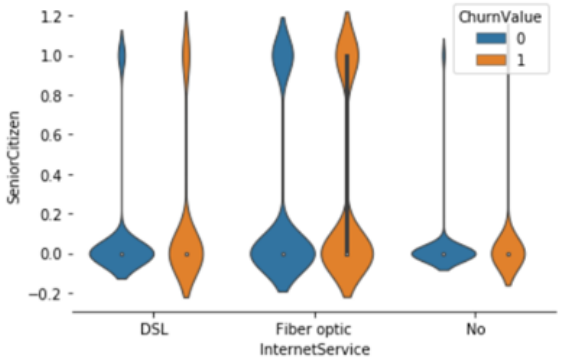


Figure 6: Correlation entre Senior Citizen et Internet Service

La fibre optique est le service Internet le plus utilisé en particulier pour les jeunes citoyens .

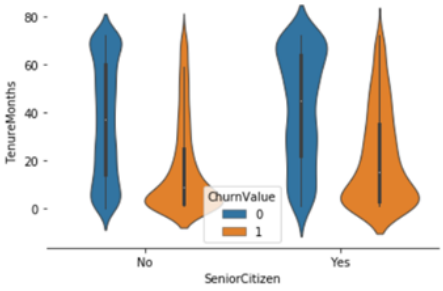


Figure 7 Correlation entre Senior Citizen et Tenure months

La figure montre que les clients jeunes et vieux avec une faible valeur de mois d'ancienneté sont beaucoup plus susceptibles de se désabonné que les clients avec des mois d'ancienneté élevés.

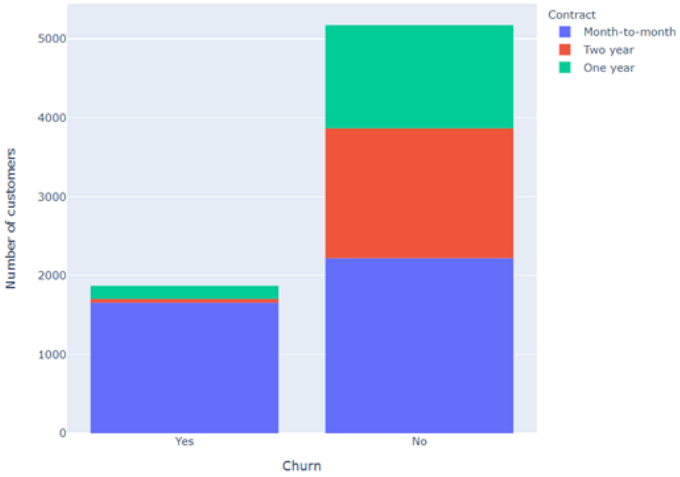


Figure 7 : Churn selon la variable Contract

Les clients avec un contrat mensuel se désabonnent dans des proportions alarmantes, tandis que les clients sous contrat de deux ans ont un taux de désabonnement moins important.

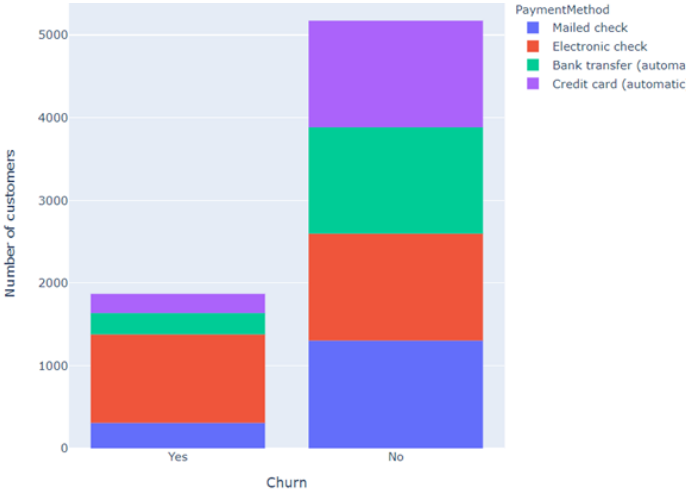


Figure 8 :Churn selon la variable Payment Method

Les clients ayant un chèque électronique comme méthode de paiement sont en proportion plus élevée (près de 50%), tandis que les autres modes de paiement en ont 15%.

PRÉPARATION DES DONNÉES

Après avoir développé la compréhension commerciale et la compréhension des données, le prochain grand objectif de la méthodologie CRISP-DM est de préparer les données pour la modélisation et l'analyse. Cela implique la sélection, le nettoyage et la transformation des données qui seront utilisées pour le projet. Bien qu'il ne s'agisse pas d'un travail tape-à-l'œil, il représente généralement 60% à 80% de l'effort d'un projet.

1.Sélection des données

Après avoir compris notre activité et nos données, nous constatons que ces variables ci-dessous sont toutes non significatives pour le problème de désabonnement.La première étape de la préparation des données consiste donc à supprimer ces colonnes: Customer ID, Count ,Country, State, City, ZipCode, Lat Long, Latitude, Longitude, churn label, CLTV, Churn Reason, Paperless Billing.

2.Nettoyage des données

2.1.Suppression des valeurs maquantes

```
ds.isnull().sum()
```

```
SeniorCitizen        0
Partner              0
Dependents           0
TenureMonths         0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
ChurnValue           0
ChurnScore           0
Male                 0
One year             0
Two year             0
dtype: int64
```

Après la conversion de type de total Charges de object en numérique nous avons remarqué qu'il y a 11 valeurs manquantes, donc on doit les éliminer..

2.2.Suppression des valeurs aberrantes

```
Q1 = ds['MonthlyCharges'].quantile(0.25)
Q3 = ds['MonthlyCharges'].quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.
filter1 = (ds['MonthlyCharges'] >= Q1 - 1.5 * IQR) & (ds['MonthlyCharges'] <= Q3 + 1.5 *IQR)
ds=ds.loc[filter1]
```

```
Q1 = ds['TenureMonths'].quantile(0.25)
Q3 = ds['TenureMonths'].quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.
filter1 = (ds['TenureMonths'] >= Q1 - 1.5 * IQR) & (ds['TenureMonths'] <= Q3 + 1.5 *IQR)
ds=ds.loc[filter1]
```

```
Q1 = ds['InternetService'].quantile(0.25)
Q3 = ds['InternetService'].quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.
filter1 = (ds['InternetService'] >= Q1 - 1.5 * IQR) & (ds['InternetService'] <= Q3 + 1.5 *IQR)
ds=ds.loc[filter1]
```

```
Q1 = ds['MonthlyCharges'].quantile(0.25)
Q3 = ds['MonthlyCharges'].quantile(0.75)
IQR = Q3 - Q1    #IQR is interquartile range.
filter1 = (ds['MonthlyCharges'] >= Q1 - 1.5 * IQR) & (ds['MonthlyCharges'] <= Q3 + 1.5 *IQR)
ds=ds.loc[filter1]
```

3.Encodage des données

```
# Categorical boolean mask
categorical_feature_mask = ds.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols = ds.columns[categorical_feature_mask].tolist()
categorical_cols
```

```
#LabelEncoder
#import labelencoder
from sklearn.preprocessing import LabelEncoder
# instantiate labelencoder object
le = LabelEncoder()
# apply le on categorical feature columns
ds[categorical_cols] = ds[categorical_cols].apply(lambda col: le.fit_transform(col))
ds.head(10)
```

Nous avons choisi d'utiliser la méthode LABEL ENCODER afin de convertir les données catégorielles en données numériques puisque les algorithmes de classification que nous avons utilisé ne prennent en compte que des variables quantitatives

MODÉLISATION

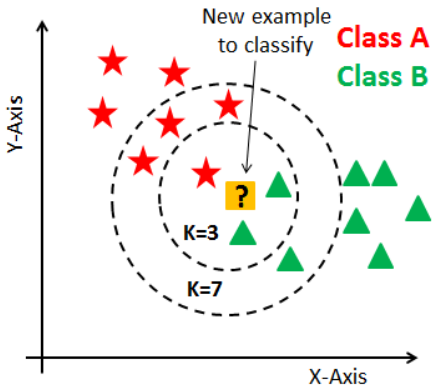
Apprentissage supervisé

1. METHODE DE CLASSIFICATION

L'objectif de la classification supervisée est principalement de définir des règles permettant de classer des objets dans des classes à partir de variables qualitatives ou quantitatives caractérisant ces objets.
Pour cela nous allons utiliser des algorithmes de classification.

1.1 k-nearest neighbors algorithm(KNN)

l'algorithme K-NN (K-nearest neighbors) est une méthode d'apprentissage supervisé. Il peut être utilisé aussi bien pour la régression que pour la classification. Son fonctionnement peut être assimilé à l'analogie suivante "dis moi qui sont tes voisins, je te dirais qui tu es…".
Pour appliquer ce modèle il est nécessaire d'avoir des données labelisées.
Les principales étapes de cet algorithmes sont :
- Séparation de la variable cibles des variables explicatives.
- Division des données en données d'apprentissage et données de test.
- Centrage réduction des variables.
- L'apprentissage .



Solutions proposés

Problème 1 : Déséquilibre des données.
Solution : Après l'exploration des données on a remarqué qu'on a un déséquilibre au niveau de la variable cible comme indiqué la figure 1 donc pour résoudre ce problème on a proposé d'utiliser l'Over Simpling et l'Under Simpling et après on compare entre les deux résultats fournis par ces deux méthodes .
Problème 2 : Déterminer les meilleurs paramètres de KNN (K : nombre des voisins, leaf size et la méthode de calcul de distance)
Solution : On a choisit d'utiliser le GridSearchCV

Première expérience :

On a divisé nos données en 70% de données d'apprentissages et 30% de données de test et on a utilisé l'oversampling pour résoudre le problème de déséquilibre entre les données .

```
clusters = RandomOverSampler(sampling_strategy='minority',random_state = 1)
clusters.fit(X,y)
X_Sampled,Y_Sampled = clusters.fit_resample(X,y)
```

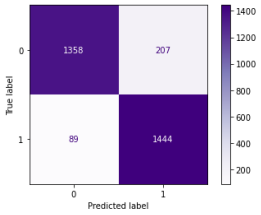
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Sampled,Y_Sampled,test_size=0.3,random_state=1)
X_train.shape, X_test.shape
```

Résultat de GridSearchCV :

```
best_parameters = clf.best_params_
print(best_parameters)
```

```
{'leaf_size': 1, 'n_neighbors': 1, 'p': 2}
```

Evaluation :
Après la réalisation de l'apprentissage avec les paramètres déterminés par le GridSearchCV on trouvait que la précision du classificateur KNN sur l'ensemble des données d'entraînement est de 100% et la précision des données de test est de 90% lorsque k = 1 (ce qui doit être évité), leaf size = 1 et p = 2 (méthode éculidienne).
F1 score = 91%
Accuracy=90%



Matrice de confusion

	precision	recall	f1-score	support
0	0.94	0.87	0.90	1565
1	0.87	0.94	0.91	1533
accuracy			0.90	3098
macro avg	0.91	0.90	0.90	3098
weighted avg	0.91	0.90	0.90	3098

Deuxième expérience

On a divisé nos données en 70% de données d'apprentissages et 30% de données de test et on a utilisé l'undersampling pour résoudre le problème de déséquilibre entre les données .

```
from imblearn.under_sampling import ClusterCentroids
clusters = ClusterCentroids(sampling_strategy='majority',random_state = 1)
clusters.fit(X,y)
X_Sampled,Y_Sampled = clusters.fit_resample(X,y)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Sampled,Y_Sampled,test_size=0.3,random_state=1)
X_train.shape, X_test.shape
```

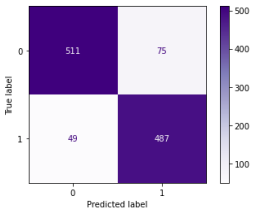
Résultat de GrisSearshCV:

```
best_parameters = clf.best_params_
print(best_parameters)
```

```
{'leaf_size': 1, 'n_neighbors': 6, 'p': 1}
```

Evaluation :

Après la réalisation de l'apprentissage avec les paramètres déterminés par le GridSearchCV on trouvait que la précision du classificateur KNN sur l'ensemble des données d'entraînement est de 94% et la précision des données de test est de 89% lorsque k = 6 , leaf size = 1 et p = 1(méthode de Manhattan).
F1 score = 89%
Accuracy=89%



Matrice de confusion

	precision	recall	f1-score	support
0	0.91	0.87	0.89	586
1	0.87	0.91	0.89	536
accuracy			0.89	1122
macro avg	0.89	0.89	0.89	1122
weighted avg	0.89	0.89	0.89	1122

Troisième expérience

On a divisé nos données en 80% de données d'apprentissages et 20% de données de test et on a utilisé l'oversampling pour résoudre le problème de déséquilibre entre les données .

```
from imblearn.under_sampling import ClusterCentroids
clusters = RandomOverSampler(sampling_strategy='minority',random_state = 1)
clusters.fit(X,y)
X_Sampled,Y_Sampled = clusters.fit_resample(X,y)
```

```
#Diviser les données (80% Apprentissage et 20% Test)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Sampled,Y_Sampled,test_size=0.2,random_state=1)
X_train.shape, X_test.shape
```

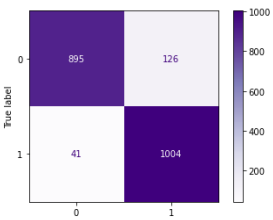
Résultat de GridSearchCV :

```
best_parameters = clf.best_params_
print(best_parameters)
```

```
{'leaf_size': 1, 'n_neighbors': 1, 'p': 2}
```

Evaluation :

Après la réalisation de l'apprentissage avec les paramètres déterminés par le GridSearchCV on trouvait que la précision du classificateur KNN sur l'ensemble des données d'entraînement est de 100% et la précision des données de test est de 92% lorsque k = 1(ce qui doit être évité) , leaf size = 1 et p = 2(méthode écludienne).
F1 score = 92%
Accuracy=92%



Matrice de confusion

	precision	recall	f1-score	support
0	0.96	0.88	0.91	1021
1	0.89	0.96	0.92	1045
accuracy			0.92	2066
macro avg	0.92	0.92	0.92	2066
weighted avg	0.92	0.92	0.92	2066

Quatrième expérience

On a divisé nos données en 80% de données d'apprentissages et 20% de données de test et on a utilisé l'undersampling pour résoudre le problème de déséquilibre entre les données .

```
from imblearn.under_sampling import ClusterCentroids
clusters = ClusterCentroids(sampling_strategy='majority',random_state = 1)
clusters.fit(X,y)
X_Sampled,Y_Sampled = clusters.fit_resample(X,y)
```

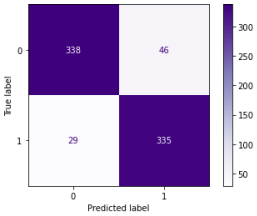
```
#Diviser les données (80% Apprentissage et 20% Test)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X_Sampled,Y_Sampled,test_size=0.2,random_state=1)
X_train.shape, X_test.shape
```

Résultat de GridSearchCV :

```
leaf_size': 1, 'n_neighbors': 8, 'p': 1}
```

Evaluation :

Après la réalisation de l'apprentissage avec les paramètres déterminés par le GridSearchCV on trouvait que la précision du classificateur KNN sur l'ensemble des données d'entraînement est de 94% et la précision des données de test est de 90% lorsque k = 8 , leaf size = 1 et p = 1(méthode de Manhattan).
F1 score = 90%
Accuracy=90%



Matrice de confusion

	precision	recall	f1-score	support
0	0.92	0.88	0.90	384
1	0.88	0.92	0.90	364
accuracy			0.90	748
macro avg	0.90	0.90	0.90	748
weighted avg	0.90	0.90	0.90	748

Conclusion

Après une comparaison entre les 4 expériences on peut constater que la quatrième expérience est la meilleure car il donne les meilleures valeurs de F1 score et accuracy.

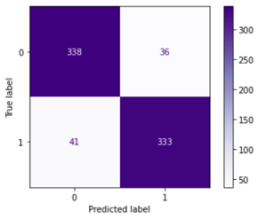
1.2 Naive Bayes

Nous allons au préalable effectuer de l'undersampling avant d'implémenter les différents algorithmes et ensuite diviser nos données en 80% pour les données d'apprentissage et 20% pour les données de test.

1.2.1 Naive Bayes sans features selection

C'est un algorithme d'apprentissage automatique supervisé qui permet de classifier un ensemble d'observations selon des règles déterminées par l'algorithme lui-même. IL est basé sur le théorème de Bayes : $$P(C\backslash X) = \frac{P(C) \times P(X\backslash C)}{P(X)}$$

où C est notre classe cible et X représente l'ensemble des variables explicatives de notre base de données.



Matrice de confusion

```
# Classification_report
from sklearn.metrics import classification_report
print(classification_report(yNB_test, yNB_pred))
```

	precision	recall	f1-score	support
0	0.89	0.91	0.90	374
1	0.91	0.89	0.90	374
accuracy			0.90	748
macro avg	0.90	0.90	0.90	748
weighted avg	0.90	0.90	0.90	748

D'après ce résultat nous voyons que l'algorithme Naive Bayes nous donne un f1-score de 90%.

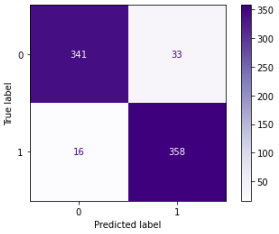
1.2.2 Naive Bayes avec le SFS feature selection

Dans cette étape nous allons avant d'effectuer l'algorithme de Naive Bayes procéder à une sélection de variables les mieux significatives .
En effet le Sequential Forward Selection commence par aucune variable et ajoute une à une en une fois la variable la plus significative sans toute fois pouvoir la retirer .
Varibles sélectionnées :

```
sfs1.k_feature_names_
```

```
('Gender',
 'InternetService',
 'OnlineBackup',
 'DeviceProtection',
 'StreamingTV',
 'StreamingMovies',
 'PaperlessBilling',
 'PaymentMethod',
 'TotalCharges',
 'ChurnScore',
 'CLTV')
```

Résultat :



Matrice de confusion

```
# Classification_report
from sklearn.metrics import classification_report
print(classification_report(yNB_test_sfs, yNB_pred_sfs))
```

```
              precision    recall  f1-score   support

           0       0.96      0.91      0.93       374
           1       0.92      0.96      0.94       374

    accuracy                           0.93       748
   macro avg       0.94      0.93      0.93       748
weighted avg       0.94      0.93      0.93       748
```

```
sfs1.k_score_
```

```
0.9321070234113712
```

D'après ce résultat nous voyons que l'algorithme Naive Bayes associé avec le SFS feature selection nous donne un f1-score de 94%.

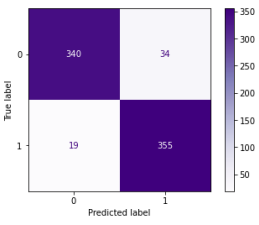
1.2.3 Naive Bayes avec le SFFS feature selection

Très similaire à la précédente à la seule différence que lorsqu'une variable est ajouté on peut la retirer.
Variables sélectionéés :

```
sffs1.k_feature_names_
```

```
('Gender',
 'Partner',
 'MultipleLines',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'PaperlessBilling',
 'PaymentMethod',
 'TotalCharges',
 'ChurnScore')
```

Résultat :



Matrice de confusion

```
from sklearn.metrics import classification_report
print(classification_report(yNB_test_sffs, yNB_pred_sffs))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       374
           1       0.91      0.95      0.93       374

    accuracy                           0.93       748
   macro avg       0.93      0.93      0.93       748
weighted avg       0.93      0.93      0.93       748
```

```
sffs1.k_score_
```

```
0.9384615384615385
```

D'après ce résultat nous voyons que l'algorithme Naive Bayes associé avec le SFFS feature selection nous donne un f1-score de 93%.

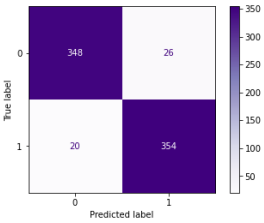
1.2.4 Naive Bayes avec le SBS feature selection

Dans cette étape nous allons avant d'effectuer l'algorithme de Naive Bayes procéder à une sélection de variables les mieux significatives .
En effet le Sequential Backward Selection commence par l'ensemble des variables et enlève une à une en une fois la variable la moin significative sans toute fois pouvoir la retirer.
Variables sélectionnées :

```
sbs1.k_feature_names_
```

```
('Gender',
 'Partner',
 'Dependents',
 'MultipleLines',
 'InternetService',
 'OnlineBackup',
 'DeviceProtection',
 'StreamingMovies',
 'PaymentMethod',
 'TotalCharges',
 'ChurnScore')
```

Résultat :



Matrice de confusion

```
from sklearn.metrics import classification_report
print(classification_report(yNB_test_sbs, yNB_pred_sbs))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94       374
           1       0.93      0.95      0.94       374

    accuracy                           0.94       748
   macro avg       0.94      0.94      0.94       748
weighted avg       0.94      0.94      0.94       748
```

```
sbs1.k_score_
```

```
0.9357859531772575
```

D'après ces observations nous voyons que l'algorithme Naive Bayes associé avec le SBS feature selection nous donne un f1-score de 94%.

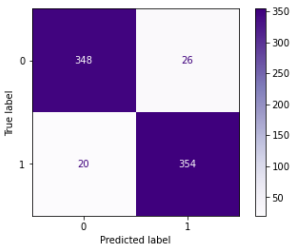
1.2.5 Naive Bayes avec le SBFS feature selection

Très similaire à la précédente à la seule différence que lorsqu'une variable est enlevée on peut la rajouter.
Variables sélectionnées :

```
sbfs1.k_feature_names_
```

```
('Gender',
 'Partner',
 'Dependents',
 'MultipleLines',
 'InternetService',
 'OnlineBackup',
 'DeviceProtection',
 'StreamingMovies',
 'PaymentMethod',
 'TotalCharges',
 'ChurnScore')
```

Résultat :



Matrice de confusion

```
from sklearn.metrics import classification_report
print(classification_report(yNB_test_sbfs, yNB_pred_sbfs))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94       374
           1       0.93      0.95      0.94       374

    accuracy                           0.94       748
   macro avg       0.94      0.94      0.94       748
weighted avg       0.94      0.94      0.94       748
```

```
sbfs1.k_score_
```

```
0.9357859531772575
```

D'après ces observations nous voyons que l'algorithme Naive Bayes associé avec le SBFS feature selection nous donne un f1-score de 94%.

Conclusion

La sélection des variables pour ces quatres différents algorithmes étant effectuée en fonction de leur différents scores, nous pouvons observer de ces derniers résultats qu'il est mieux d'appliquer l'algorithme Naive Bayes avec des algorithmes de sélection de variables car ces derniers nous donnent des résultats plus précis.

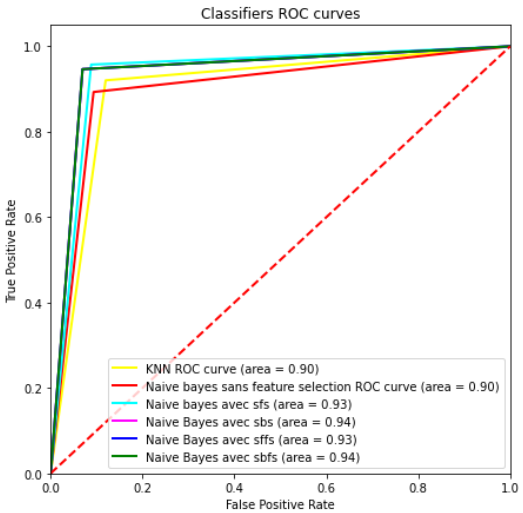
1.2.5 Comparaison entre KNN et Naive bayes

Tableau comparatif

On va afficher un tableau qui ordonne les modèles du plus performant au moins performant suivant le score

	Model	Score
3	Naive Bayes avec sbs	0.938503
5	Naive Bayes avec sbfs	0.938503
2	Naive bayes avec sfs	0.934492
4	Naive Bayes avec sffs	0.929144
0	KNN	0.899733
1	Naive bayes sans feature selection	0.899733

Courbe ROC



D'après la courbe et le tableau ci-dessus on peut constater que la naïve Bayes avec la sélection des fonctionnalités est le plus précis car il détecte les vrais positives le premier et il correspond à la valeur de F1 score le plus élevé.

1.3 Arbre De Décision

Nous allons dans ces algorithmes diviser nos données en 70% de données d'apprentissage et 30% de données de test et éffectuer un oversampling sur ces données.

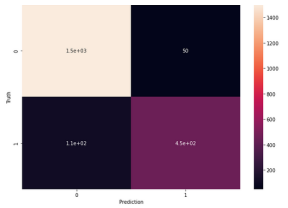
1.3.1 Random Forest

L'algorithme des « forêts aléatoires » effectue un apprentissage en parallèle sur de multiples arbres de décision construits aléatoirement et entraînés sur des sous-ensembles de données différents. Concrètement, chaque arbre de la forêt aléatoire est entrainé sur un sous ensemble aléatoire de données selon le principe du bagging, avec un sous ensemble aléatoire de variables des données selon le principe des «projections aléatoires ». Chaque arbre individuel dans la forêt aléatoire fourni une prédiction de classe et la classe avec le plus de votes devient la prédiction de notre modèle.

Dans notre travail nous avons utilisé 10 arbres et divisé nos données en 70% de données d'apprentissage et 30% de données de test.

```
: from sklearn.ensemble import RandomForestClassifier

: X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=3)

: model=RandomForestClassifier(n_estimators=10)
  model.fit(X_train, y_train)

: RandomForestClassifier(n_estimators=10)

: model.score(X_test,y_test)

: 0.9227488151658768
```

Résultat :



	precision	recall	f1-score	support
NO	0.92985723	0.96770026	0.94840139	1548
Yes	0.89979960	0.79893238	0.84637135	562
accuracy			0.92274882	2110
macro avg	0.91482842	0.88331632	0.89738637	2110
weighted avg	0.92185136	0.92274882	0.92122562	2110

Au vu de ces résultats nous obtenons avec un Random Forest à 10 arbres une précision de 92.27%.

1.3.2 ID3

ID3 construit l'arbre de décision récursivement.
A chaque étape de la récursion, il calcule parmi les attributs restant pour la branche en cours, celui qui maximisera le gain d'information.
Le calcul ce fait à base de l'entropie de Shanon.
L'algorithme suppose que tous les attributs sont catégoriels ; Si des attributs sont numériques, ils doivent etre descritisés pour pouvoir l'appliquer.

Il génère automatiquement sa matrice de confusion et ses pourcentages d'évaluation.

Résultat :

```
ID3  tree is going to be built...
-------------------------
finished in  203.57198119163513  seconds
-------------------------
Evaluate  train set
-------------------------
Accuracy:  94.8947667804323 % on  7032  instances
Labels:  ['1' '0']
Confusion matrix:  [[1665, 155], [204, 5008]]
Precision:  91.4835 %, Recall:  89.0851 %, F1:  90.2684 %
```

Nous pouvons tirer de ce résultat que l'algorithme ID3 génère par lui meme un f1-score de 90.2684% .

1.3.3 C4.5

C'est une amélioration de l'algorithme ID3,
Prend en compte les attributs numérique ainsi que les valeurs manquantes.
L'algorithme utilise la fonction du gain d'entropie combiné avec une fonction SplitInfo pour évaluer les attributs chaque itération.
Attributs discrets : Gain et permet le regroupement,
Valeurs manquante :
pour le test : prendre la classe majoritaire
pour l'entrainement prendre la distribution des valeurs connues
Il génère lui aussi automatiquement sa matrice de confusion et ses pourcentages d'évaluation.

Résultat :

```
C4.5  tree is going to be built...
-------------------------
finished in  196.9807906150818  seconds
-------------------------
Evaluate  train set
-------------------------
Accuracy:  94.55346985210467 % on  7032  instances
Labels:  ['1' '0']
Confusion matrix:  [[1681, 195], [188, 4968]]
Precision:  89.6055 %, Recall:  89.9411 %, F1:  89.773 %
```

Nous en tirons de ce résultat que l'algorithme C4.5 génère par lui meme un f1-score de 89.773% .

1.3.4 CHAID

Très similaire aux algorithmes précédent et choisit à chaque la variable indépendante dont l'interaction avec la variable dépendante est la plus forte en s'appuyant sur le test du KHI-2.
Il faut également d'après notre analyse renommer la variable cible avant de l'appliquer.
Il génère lui aussi automatiquement sa matrice de confusion et ses pourcentages d'évaluation.

Résultat :

```
CHAID  tree is going to be built...
-------------------------
finished in  213.53593468666077  seconds
-------------------------
Evaluate  train set
-------------------------
Accuracy:  94.21217292377702 % on  7032  instances
Labels:  ['1' '0']
Confusion matrix:  [[1647, 185], [222, 4978]]
Precision:  89.9017 %, Recall:  88.122 %, F1:  89.003 %
```

Nous en tirons de ce résultat que l'algorithme C4.5 génère par lui meme un f1-score de 89.003% .

1.3.5 CART

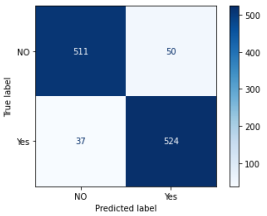
L'algorithme CART (Classification and Regression Trees) c'est l'une des méthodes d'apprentissage supervisé les plus populaires pour la classification.
Les décisions possibles sont situées aux extrémités des branches « les feuilles » , dans notre cas c'est un churner ou non et sont atteintes en fonction de décisions prises à chaque étape par la maximisation de l'indice de GINI.

Pour appliquer cet algorithme on a commencé par la séparation entre les variables explicatives et la valeur cible «ChurnValue » , ensuite comme le pourcentage des chuners et non churners n'est pas équilibré pour éviter le risque d'un overfiting on a appliquer le oversampling sur la classe minoritaire .
Pour cette arbre les données sont diviser 70% pour l'apprentissage et 30% pour le test. Les autres paramètres comme le Critère de séparation et max_depth on a appliqué GridSearch pour tester toutes le combinaison possibles pour avoir la meilleur solution avec une arbre plus lisible et performances très élevés.

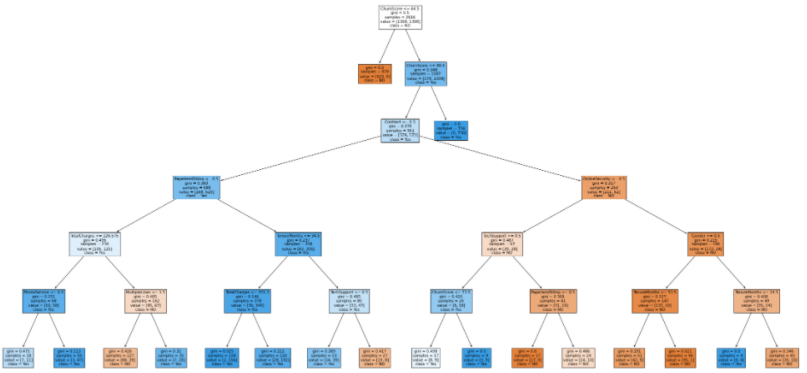
```
grid.best_params_
```

```
{'criterion': 'gini', 'max_depth': 7}
```

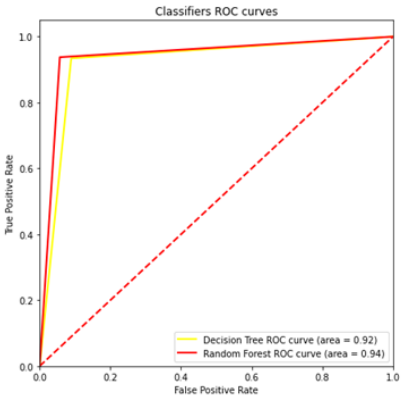
Résultat :



	precision	recall	f1-score	support
NO	0.93248175	0.91087344	0.92155095	561
Yes	0.91289199	0.93404635	0.92334802	561
accuracy			0.92245989	1122
macro avg	0.92268687	0.92245989	0.92244948	1122
weighted avg	0.92268687	0.92245989	0.92244948	1122



Courbe ROC



D'après la courbe ci-dessus on peut constater que le Random Forest est l'algorithme le plus précis car il détecte les vrais positives le premier et il correspond a la valeur de F1 score le plus élevé .

APPRENTISSAGE NON SUPERVISE

1. CLUSTERING

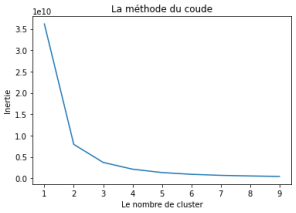
C'est une méthode en analyse des données. Elle vise à diviser un ensemble de données en différents « paquets » homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité (similarité informatique) que l'on définit en introduisant des mesures et classes de distance entre objets.

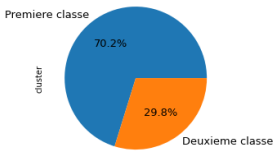
1.1 K-MEANS

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents. La difficulté résidera dans le fait de déterminer le nombre k mais dans notre travail nous le déterminerons à l'aide de la méthode du coude et il est évident que dansnotre cas k=2.

```
#Détermination du nombre de cluster grace à la methode du coude
inertie=[]
for i in range(1,10):
    kmeans=KMeans(n_clusters=i)
    kmeans.fit(dt1)
    inertie.append(kmeans.inertia_)
```

```
plt.plot(range(1,10),inertie)
plt.title("La méthode du coude")
plt.xlabel("Le nombre de cluster")
plt.ylabel("Inertie")
plt.show()
```





Résultat

col_0	0	1
ChurnValue		
0	3395	1768
1	1548	321

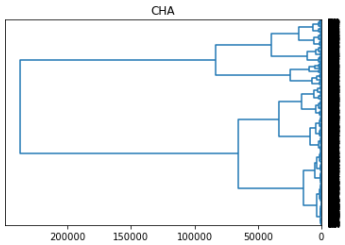
1.2 CAH

Le principe de la CAH est de rassembler des individus selon un critère de ressemblance défini au préalable qui s'exprimera sous la forme d'une matrice de distances, exprimant la distance existant entre chaque individu pris deux à deux. Deux observations identiques auront une distance nulle. Plus les deux observations seront dissemblables, plus la distance sera importante. La CAH va ensuite rassembler les individus de manière itérative afin de produire un dendrogramme ou arbre de classification.

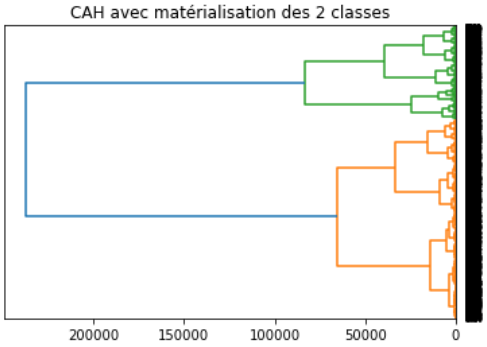
Nous avons donc généré la matrice des distances avec la métohde Ward en utilisant la métrique Euclidienne et afficher le dendogramme.

```
#générer la matrice des distances
Z = linkage(dt1,method='ward',metric='euclidean')
```

```
#affichage du dendrogramme
plt.title("CHA")
dendrogram(Z,labels=dt1.index,orientation='left',color_threshold=0)
plt.show()
```



Ensuite nous avons ainsi matérialisé les les deux classes.



Résultat

col_0	1	2
ChurnValue		
0	3282	1881
1	1512	357

CONCLUSION

Ce projet nous a également permis de consolider nos connaissances en Machine Learning .
Il nous a permis de mettre en application et de nous familiariser avec certaines notions de machine learning étudiées comme la méthodologie CRISP qui nous a permis de structurer notre projet.
On a aussi déployé plusieurs systèmes Machine learning (supervised and unsupervised methods) comme : celles de classification (k-NN , Naïve bayes sans et avec features selection , Decision tree) et celles de segmentation (Clustering): K-means.
Enfin, ce projet nous permettra donc d'être plus performants sur d'éventuels travaux futurs notamment le deep learning et ses techniques comme Classic Neural Networks etc…