# Python

Arbre de décision avec la bibliothèque scikit-learn

# Préparation des données (contraintes)

- ✓ Pas de valeurs manquantes

  S'il y en a, une méthode classique est de les remplacer : on parle d'**imputation**. Cette imputation peut se faire par la moyenne, la médiane, la valeur modale (la plus fréquente)…

- ✓ Tableau de **nombres** (**array numpy**),

  Ce qui signifie que les variables qualitatives doivent être codées numériquement.

# Transformation en tableau de **nombres**

## Recodage des variables qualitatives

```
#Recodage de 'X2'(ville)
#Pour recoder facilement la variable, on commence par la transformer
#en variable catégorielle
exemple["X2"]=pd.Categorical(exemple["X2"],ordered=False)

#Affichage des modalités
exemple["X2"].unique()
#--> ['Bayonne', 'Anglet', 'Biarritz']

#Recodage
exemple["X2"] = exemple["X2"].cat.rename_categories({'Bayonne': 1,
                                                     'Biarritz': 2,
                                                     'Anglet': 3})
```

# Transformation en tableau de **nombres**

## Extraction des données vers le tableau et de la cible

```
#Création du tableau des variables explicatives (.values pour transformer
#le dataframe en tableau numpy (array)
x = exemple.iloc[:,0:3].values

#Création de la cible
cible = exemple['Y']
```

# Paramétrage et création de l’arbre

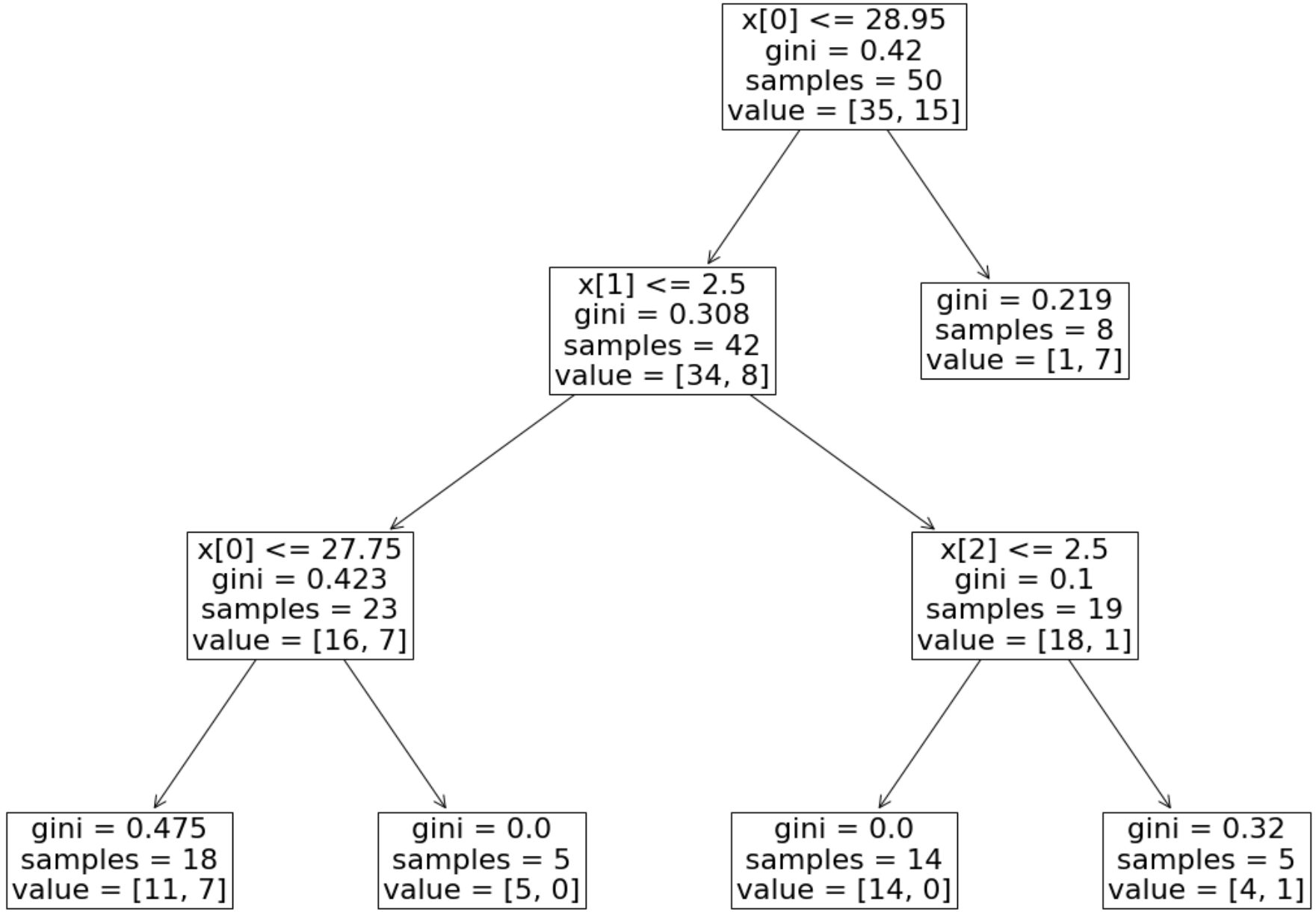
```
#Paramétrage de l’arbre
#Profondeur max : 3, taille minimum des feuilles : 5
class_arbre = DecisionTreeClassifier(max_depth = 3, min_samples_leaf=5)
```

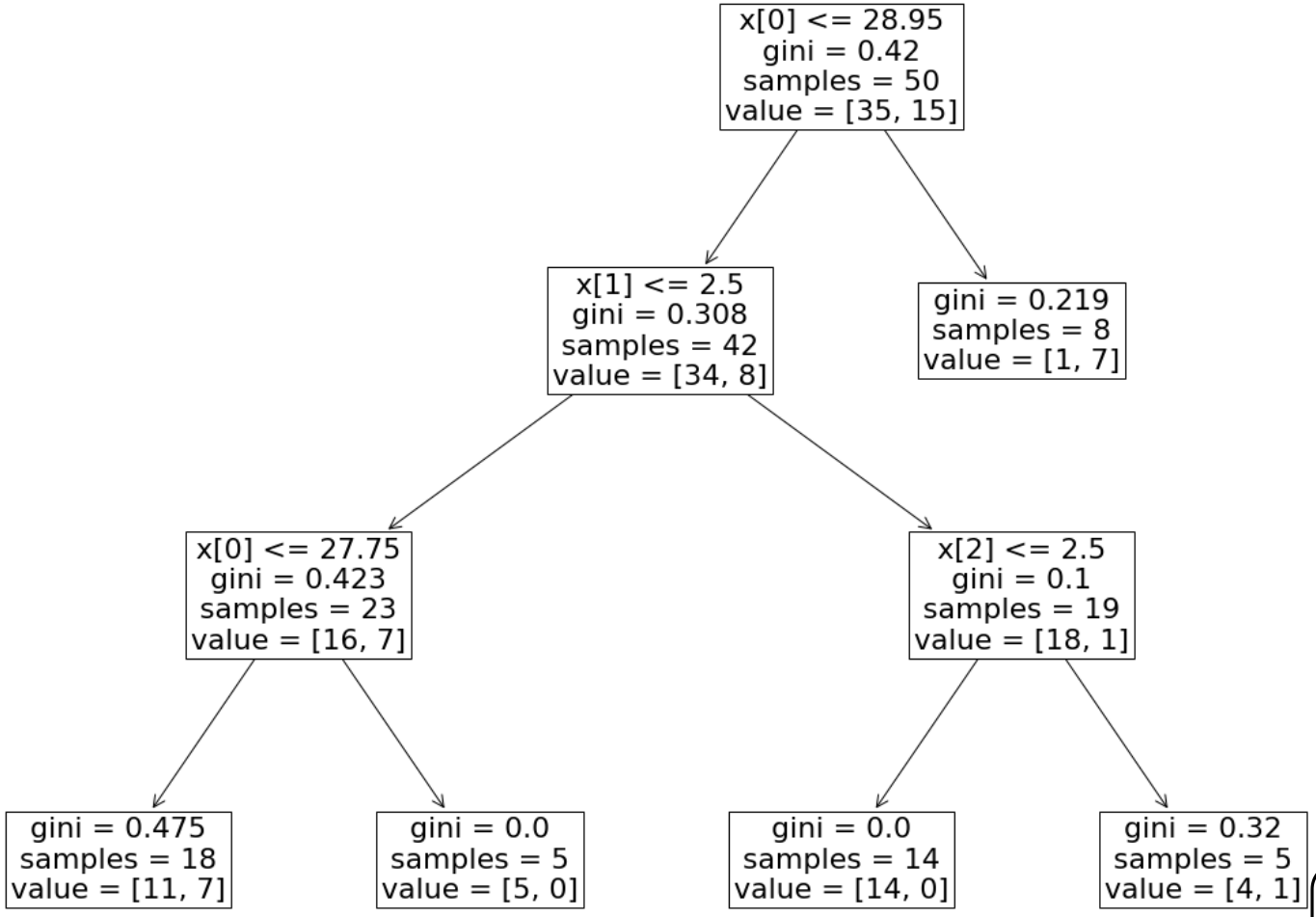
# Paramétrage et création de l’arbre

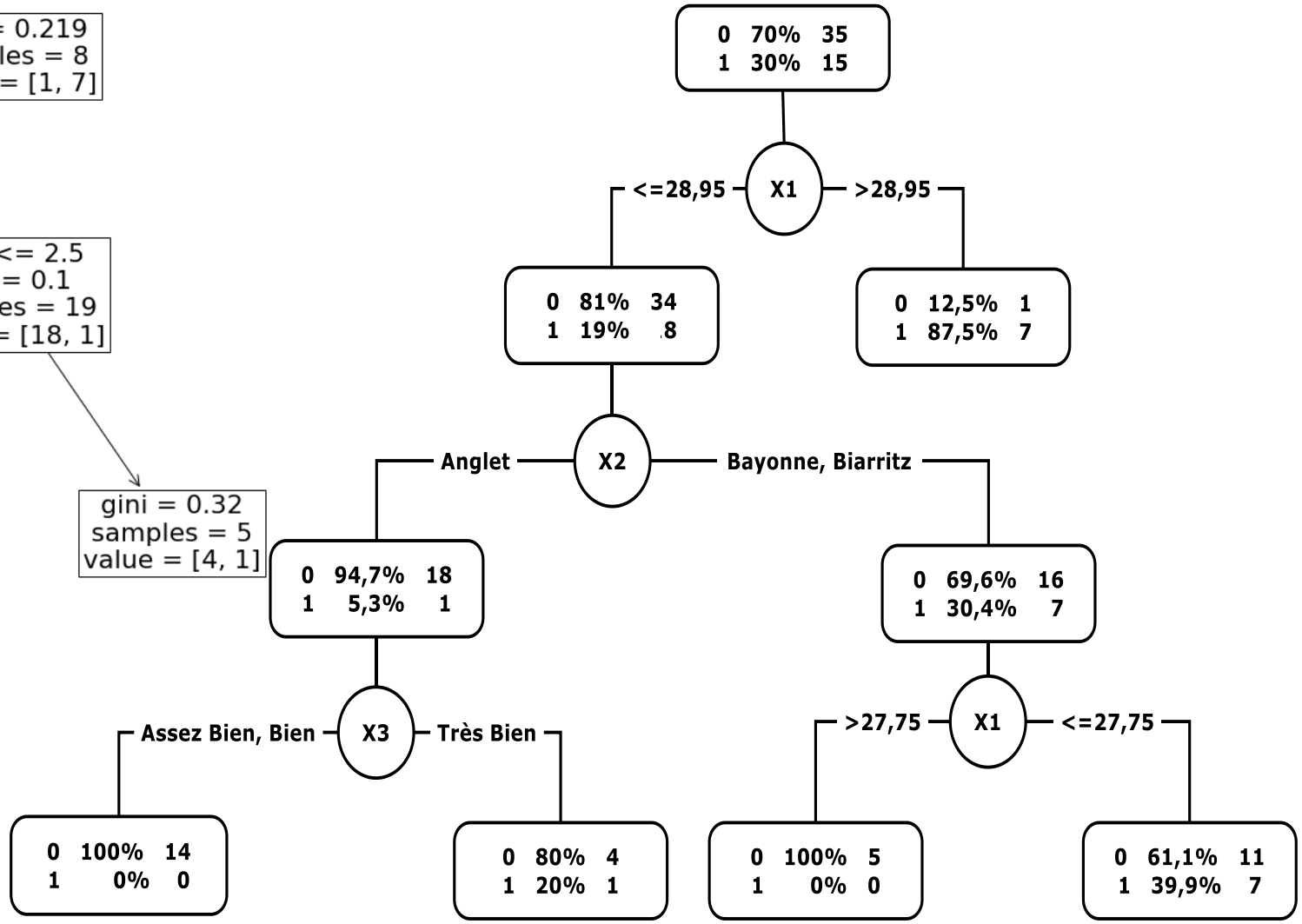
```
#Paramétrage de l’arbre
#Profondeur max : 3, taille minimum des feuilles : 5
class_arbre = DecisionTreeClassifier(max_depth = 3, min_samples_leaf=5)

#Calcul de l’arbre dans digit_free
digit_tree = class_arbre.fit(x, cible)
```

# Paramétrage et création de l’arbre

```
#Paramétrage de l’arbre
#Profondeur max : 3, taille minimum des feuilles : 5
class_arbre = DecisionTreeClassifier(max_depth = 3, min_samples_leaf=5)

#Calcul de l’arbre dans digit_free
digit_tree = class_arbre.fit(x, cible)

#Visualisation de l’arbre
plt.figure(figsize = (20,15))
tree.plot_tree(digit_tree, fontsize = 22)
plt.show()
```


x[0] <= 28.95
gini = 0.42
samples = 50
value = [35, 15]
x[1] <= 2.5
gini = 0.308
samples = 42
value = [34, 8]
gini = 0.219
samples = 8
value = [1, 7]
x[0] <= 27.75
gini = 0.423
samples = 23
value = [16, 7]
x[2] <= 2.5
gini = 0.1
samples = 19
value = [18, 1]
gini = 0.475
samples = 18
value = [11, 7]
gini = 0.0
samples = 5
value = [5, 0]
gini = 0.0
samples = 14
value = [14, 0]
gini = 0.32
samples = 5
value = [4, 1]


x[0] <= 28.95
gini = 0.42
samples = 50
value = [35, 15]
x[1] <= 2.5
gini = 0.308
samples = 42
value = [34, 8]
gini = 0.219
samples = 8
value = [1, 7]
x[0] <= 27.75
gini = 0.423
samples = 23
value = [16, 7]
x[2] <= 2.5
gini = 0.1
samples = 19
value = [18, 1]
gini = 0.475
samples = 18
value = [11, 7]
gini = 0.0
samples = 5
value = [5, 0]
gini = 0.0
samples = 14
value = [14, 0]
gini = 0.32
samples = 5
value = [4, 1]


0 70% 35
1 30% 15
X1
<=28,95
>28,95
0 81% 34
1 19% 8
0 12,5% 1
1 87,5% 7
X2
Anglet
Bayonne, Biarritz
0 94,7% 18
1 5,3% 1
0 69,6% 16
1 30,4% 7
X3
X1
Assez Bien, Bien
Très Bien
>27,75
<=27,75
0 100% 14
1 0% 0
0 80% 4
1 20% 1
0 100% 5
1 0% 0
0 61,1% 11
1 39,9% 7

# Remarques

Les variables **nominales** ne sont pas correctement traitées, en effet, les modalités ne peuvent être regroupées que par valeurs consécutives, comme s'il s'agissait de variables qualitatives ordinales.

La seule solution est d'associer à chaque variable nominale autant de variables binaires que de modalités (codage disjonctif complet):

| Ville | Avis | Cible |
|---|---|---|
| **Bayonne** | Très Bien | 1 |
| **Anglet** | Bien | 1 |
| **Biarritz** | Assez Bien | 0 |
| **Anglet** | Assez Bien | 0 |
| **Biarritz** | Bien | 0 |
| **Bayonne** | Très Bien | 0 |
| **Anglet** | Très Bien | 1 |
| **Bayonne** | Bien | 1 |
| **Bayonne** | Assez Bien | 0 |
| **Anglet** | Bien | 0 |
| **Biarritz** | Assez Bien | 0 |
| **Anglet** | Très Bien | 1 |
| **Bayonne** | Bien | 1 |

| Bayonne | Anglet | Biarritz | Avis | Cible |
|---|---|---|---|---|
| **1** | 0 | 0 | Très Bien | 1 |
| **0** | 1 | 0 | Bien | 1 |
| **0** | 0 | 1 | Assez Bien | 0 |
| **0** | 1 | 0 | Assez Bien | 0 |
| **0** | 0 | 1 | Bien | 0 |
| **1** | 0 | 0 | Très Bien | 0 |
| **0** | 1 | 0 | Très Bien | 1 |
| **1** | 0 | 0 | Bien | 1 |
| **1** | 0 | 0 | Assez Bien | 0 |
| **0** | 1 | 0 | Bien | 0 |
| **0** | 0 | 1 | Assez Bien | 0 |
| **0** | 1 | 0 | Très Bien | 1 |
| **1** | 0 | 0 | Bien | 1 |

# Remarques : pb variables nominales

Les variables **nominales** ne sont pas correctement traitées, en effet, les modalités ne peuvent être regroupées que par valeurs consécutives, comme s'il s'agissait de variables qualitatives ordinales.

La seule solution est d'associer à chaque variable nominale autant de variables binaires que de modalités (codage disjonctif complet):

| Ville | Avis | Cible |
|---|---|---|
| **Bayonne** | Très Bien | 1 |
| **Anglet** | Bien | 1 |
| **Biarritz** | Assez Bien | 0 |
| **Anglet** | Assez Bien | 0 |
| **Biarritz** | Bien | 0 |
| **Bayonne** | Très Bien | 0 |
| **Anglet** | Très Bien | 1 |
| **Bayonne** | Bien | 1 |
| **Bayonne** | Assez Bien | 0 |
| **Anglet** | Bien | 0 |
| **Biarritz** | Assez Bien | 0 |
| **Anglet** | Très Bien | 1 |
| **Bayonne** | Bien | 1 |

| Bayonne | Anglet | Biarritz | Avis | Cible |
|---|---|---|---|---|
| **1** | 0 | 0 | Très Bien | 1 |
| **0** | 1 | 0 | Bien | 1 |
| **0** | 0 | 1 | Assez Bien | 0 |
| **0** | 1 | 0 | Assez Bien | 0 |
| **0** | 0 | 1 | Bien | 0 |
| **1** | 0 | 0 | Très Bien | 0 |
| **0** | 1 | 0 | Très Bien | 1 |
| **1** | 0 | 0 | Bien | 1 |
| **1** | 0 | 0 | Assez Bien | 0 |
| **0** | 1 | 0 | Bien | 0 |
| **0** | 0 | 1 | Assez Bien | 0 |
| **0** | 1 | 0 | Très Bien | 1 |
| **1** | 0 | 0 | Bien | 1 |

# Remarques : imputation valeurs manquantes

## Exemple données Titanic

```
#Exemple sur les données du titanic
df=pd.read_table('titanic.csv',sep = ',',
                 header=0,usecols=[1,2,4,5,9,11],
                 names=["Surv","Classe","Genre","Age","Prix","Port"],

#Pour vérifier si les données ont des valeurs manquantes, on peut utilise
#l'instruction count, qui dénombre les valeurs non manquantes de chaque colonne
df.count()
# Résultat :
#       Surv       891
#       Classe     891
#       Genre      891
#       Age        714
#       Prix       891
#       Port       889
# Il y a donc des valeurs manquantes dans Age (quanti) et Port (quali)
```

# Remarques : imputation valeurs manquantes

## Exemple données Titanic

```
# Imputation des valeurs manquantes :

# Par la médiane pour Age
df["Age"]=df["Age"].fillna(df["Age"].median())


# Par le mode pour Port.
# Attention, comme il peut y avoir plusieurs valeurs modales,
# la méthode mode() ne renvoie pas une valeur mais une  (on décide de prendre
# la première valeur de cette série
df["Port"]=df["Port"].fillna(df["Port"].mode()[0])
```

# Travaux pratiques

Le fichier attrition.csv qui comprend 21 variables et 3333 individus, clients d'un opérateur de téléphonie mobile. L'objectif est de prévoir les départs vers un autre opérateur.

| | |
|---|---|
| **etat** | État (adresse) du client |
| **Mois_anc** | Ancienneté du compte, en mois |
| **dept** | Département du client |
| **tel** | Numéro de téléphone |
| **international** | Option pour les appels à l'international (oui/non) |
| **messag_voc** | Option pour la messagerie vocale (oui/non) |
| **Nb_message** | Nombre de messages sur la messagerie vocale |
| **Min_jour** | Nombre total de minutes de consommation dans la journée |
| **Appels_jour** | Nombre total d'appels dans la journée |
| **CA_jour** | Chiffre d'affaire total dans la journée |
| **Min_soir** | Nombre total de minutes de consommation en soirée |
| **Appels_soir** | Nombre total d'appels en soirée |
| **CA_soir** | Chiffre d'affaire total en soirée |
| **Min_nuit** | Nombre total de minutes de consommation la nuit |
| **Appels_nuit** | Nombre total d'appels la nuit |
| **CA_nuit** | Chiffre d'affaire total la nuit |
| **Min_inter** | Nombre total de minutes d'appels à l'international |
| **Appels_inter** | Nombre total d'appels à l'international |
| **CA_inter** | Chiffre d'affaire total à l'international |
| **Appels_servcli** | Nombre d'appels au service clientèle de l'opérateur |
| **attrition** | Indique si le client a quitté l'opérateur (true) |