REPUBLIQUE DU SENEGAL

Un peuple-Un but-Une foi

****

MINISTERE DE L'ECONOMIE, DU PLAN ET DE LA COOPERATION

****

(ANSD)

ANSD
Agence Nationale de la Statistique et de la Démographie

AGENCE NATIONALE DE LA STATISTIQUE ET DE LA DEMOGRAPHIE

****

ECOLE NATIONALE DE LA STATISTIQUE ET DE L'ANALYSE ECONOMIQUE
Pierre Ndiaye (ENSAE)

PROJET DE FIN DE MODULE DE DATA MINING EN AS3

PROJET DE DATA MINING : ANALYSE, TRAITEMENT, CLASSIFICATION ET RECHERCHE

Rédigé par :

DEKOU VESSOU Jeff-Maruis

SOUS LA SUPERVISION DE :
M. GOMIS

ANNEE ACADEMIQUE 2021-2022

PARTIE I : TRAVAUX PRATIQUES DE DATAMINING

I. INTRODUCTION

Le but de cet exercice consiste à créer un modèle de prédiction de la survie des passagers suite à son naufrage ou accident. C'est également une application concrète et complète de la classification, du traitement sur un jeu de donnée réel. Le jeu de données contient initialement les données de 891 passagers, réparties sur 12 variables. En quelques lignes, notre travail ici consistera à :

- Récupérer et explorer les données
- Préparer / mettre en forme / réparer les données
- Récupérer ou créer des données supplémentaires pour aider les algorithmes
- Choix des variables les plus pertinentes
- La mise en place de l'arbre de décision et du modèle
- Evaluation de la performance du modèle

1. *Premier regard sur les données et signification des variables :*

Nos données sont recueillies dans la base de données « base.csv ». Dans cette base, chaque passager possède un identifiant allant de 1 à 891.

Tableau 1: Présentation des données

Variables	Description	Notes sur les valeurs
PassengerID	Identifiant du passager	Entier compris entre 1 et 1309
Survived	Survivant ?	1 si le passager a survécu, 0 s'il est décédé
Pclass	Classe du passager	1 = 1ère classe, 2 = 2ème classe, 3 = 3ème classe
Name	Nom et titre du passager	Style : Nom, titre. Prénoms
Sex	Sexe du passager	'male' ou 'female'
Age	Age du passager	Décimal si inférieur à 1, estimé si de la forme xx.5
SibSp	Nombre d'époux, de frères ou de sœurs présents à bord	
Parch	Nombre de parents ou d'enfants présents à bord	
Ticket	Numéro du ticket	
Fare	Prix des tickets	Le prix est indiqué en £ et pour un seul achat (peut correspondre à plusieurs tickets)
Cabin	Numéro de Cabine	Un ou plusieurs numéros de cabine, de la forme 'A123'
Embarked	Port d'embarcation	C, Q S

2. *Installation et importation des packages*

Pour une bonne analyse sur Python, il nous faut un certain nombre de packages. On peut énumérer entre autres :

- Pandas : traitement (manipulation, analyse) des tableaux de données mixtes (variables numériques, textuelles, booléens ..)
- Numpy : manipulation des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- SciKit Learn : librairie destinée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme l'Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment NumPy et SciPy.
- Et pour la visualisation : matplotlib et seaborn

Tableau 2: Chargement des Librairies Python

```
# Pandas pour manipuler les tableaux de données
import pandas as pd


# Numpy pour les listes de données numériques et les fonctions classiques mathématiques
import numpy as np

# scipy (librairie scientifique) pour les fonctions statistiques et autres utilisaires
import scipy

# scikit learn pour les outils de datamining et prediction
import sklearn
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler
from scipy.stats.contingency import association
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz

# librairies pour la visualisation de données
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
```

3. *Importations de notre base CSV :*

On importe notre base CSV.

```
#%%Importation de la base zt visualisation
data = pd.read_csv('C:/Users/DELL/Desktop/Projets_AS3/projet/projet/base.csv')
```

4. *Visualisation générale de la base*

Comme nous avons importés la base, nous allons faire ressortir une vue d'ensemble sur notre base en ce qui concerne la taille les variables et leur nombre de données valides et manquantes.

```
print(data.shape)    # dimension du tableau (891;12)
print(data)
```

Nous allons présenter de façon globale nos données

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
..                                                ...     ...   ...    ...
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
```

Nous allons également nous intéresser à la visualisation des variables.

```
print(data.columns)
data.isnull().sum()
```

Cette visualisation nous montre des résultats suivants :

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')

Out[7]: PassengerId      0
        Survived         0
        Pclass           0
        Name             0
        Sex              0
        Age            177
        SibSp            0
        Parch            0
        Ticket           0
        Fare             0
        Cabin          687
        Embarked         2
        dtype: int64
```

On peut voir que les variables Age, Cabin et Embarked présentent des valeurs manquantes. Nous allons par la suite expliciter la manière donc nous nous mettrons pour traiter ces données aux valeurs manquantes.

Néanmoins nous devons retenir que le traitement ne peut se faire que sur les variables pertinentes que nous choisirons dans la suite de nos analyses. La partie suivante s'axera alors le choix de nos variables pertinentes, les critères de ce choix, les traitements des valeurs manquantes ou aberrantes s'il a lieu et allons également faire ressortir les motifs du rejet de certains variables.

II. TRAITEMENT ET CHOIX DES VARIABLES

Dans cette partie, nous allons mettre l'accent sur les critères du choix de nos variables ; nos variables choisies sans oublier de faire ressortir les différents traitements effectués au niveau de ces variables.

1. *Critère du choix des variables pertinentes de notre modèle*

Pour faire le choix de nos variables, nous avons procédé de façon graphique par une visualisation de l'association de notre variable et la variable cible (Survived dans notre cas) d'une part et d'autre part on mesure la corrélation entre les variables qualitatives à l'aide du coefficient de Cramer. Pour certaines variables on va même jusqu'à faire les tableaux croisés.

❖ Visualisation graphique

Cette approche nous permet de représenter l'association de nos variables et de la variable cible. Elle nous aide à faire ressortir les groupes ayant plus survécus ou non. Néanmoins pour confirmer cette hypothèse on s'appuie sur le coefficient de cramer que nous avons construit.

❖ Coefficient de Cramer

✓ *Un peu de théorie*

Contrairement au χ^2, il reste stable si l'on augmente la taille de l'échantillon dans les mêmes proportions intermodalités. Il est basé sur le χ^2 maximal que le tableau de contingence pourrait théoriquement produire : ce dernier aurait alors une seule case non nulle par ligne ou par colonne (selon que le tableau a plus de lignes ou plus de colonnes). Ce χ^2 max théorique est égal à l'effectif multiplié par le plus petit côté du tableau (nombre de lignes ou de colonnes) moins 1. Par exemple un tableau de 2 × 3 avec un effectif de 100 a pour χ^2 max 100 × (2 – 1) = 100.

Le V de Cramer est la racine carrée du χ^2 divisé par le χ^2 max.

$$V = \sqrt{\frac{\chi^2}{\chi^2_{max}}} = \sqrt{\frac{\chi^2}{n \times [\min(l,c) - 1]}}$$

V de Cramer

Plus V est proche de zéro, plus il y a indépendance entre les deux variables étudiées. Il vaut 1 en cas de complète dépendance puisque le χ^2 est alors égal au χ^2 max (dans un tableau 2 × 2, il prend une valeur comprise entre -1 et 1).

- ✓ *Mise en œuvre sous python*

```
#%%FONCTION CRAMER
# cette fonction évalue la corrélation entre variables qualitatives en
# - élaboration du tableau de contingence des valeurs
# - calcul du chi2 de cet tableau
# - calcul du coefficient de cramer qui est une normalisation du coefficient chi2

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x,y)
    chi2 = scipy.stats.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2/n
    r,k = confusion_matrix.shape
    phi2corr = max(0, phi2-((k-1)*(r-1))/(n-1))
    rcorr = r-((r-1)**2)/(n-1)
    kcorr = k-((k-1)**2)/(n-1)
    return np.sqrt(phi2corr/min((kcorr-1),(rcorr-1)))
```

Des fois pour une bonne visualisation on associe les tableaux croisés aux critères. Sur la base de ces critères nous sommes parvenus à choisir 7 variables dont Une variable créée (Title qui est le titre du passager) et 6 variables locales de notre base : La classe du passager (Pclass), l'âge (Age), le Sexe (Sex), le lieu d'embarcation (Embarked), le nombres de frères et sœur à bord (SibSp) et les parents et enfants (Parch). La partie suivante mettra la lumière sur les raisons du choix de ces variables.

2. Traitement des données et raison du choix de nos variables

Dans cette partie nous allons faire ressortir tous les variables, leur traitement et à partir des critères, faire le choix.

a. Survived : la variable cible

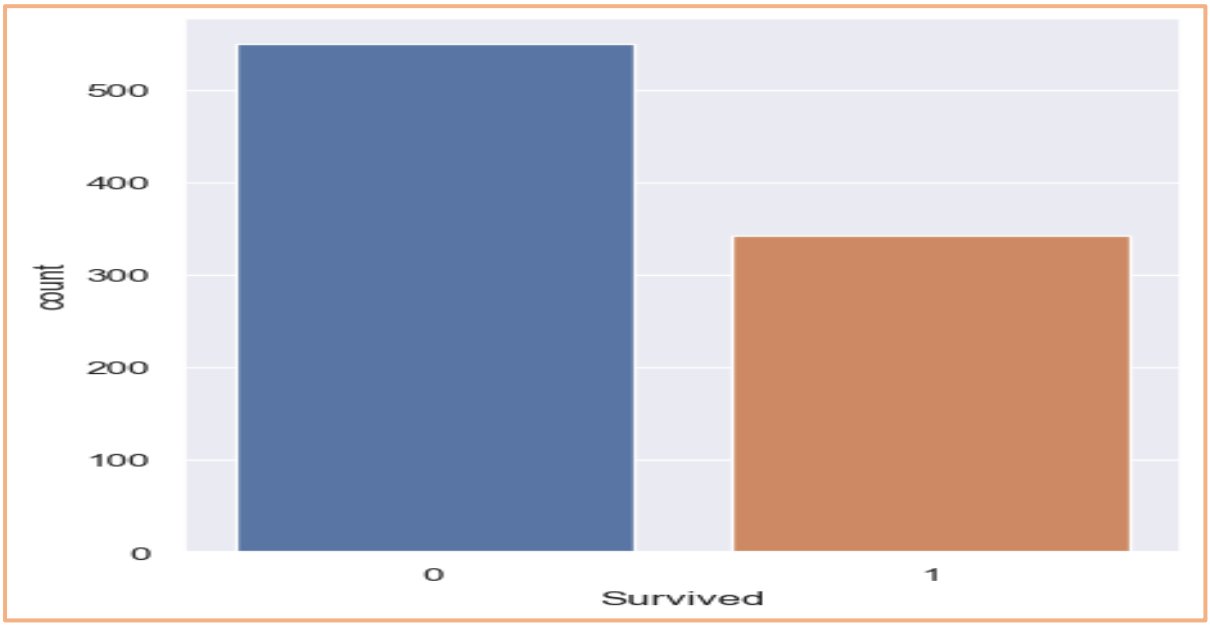
Notre objectif étant de prédire la survie des passagers, il est important avant de commencer notre analyse de visualiser cette variable qu'est la survie.

```
Entrée [17]: #Visualisation variable survived
             data.Survived.value_counts()

Out[17]: 0    549
         1    342
         Name: Survived, dtype: int64
```

Ce résultat nous montre que les données sont numériques (0 pour les passagers disparus, 1 pour les survivants), il n'y a pas de valeurs manquantes. Il n'y a pas également de valeurs aberrantes comme nous le montre visuellement à partir du code ci-dessus ; la répartition des survivants et des victimes :

```
sns.set(rc={'figure.figsize':(5,6)})#figursize pour la largeur et la hauteur de notre plot
_ = sns.countplot(x="Survived", data=data)
```

Figure 1:Repartition des survivants et des victimes



On peut voir clairement qu'il a plus de victimes que de survivants soit 62% de victimes contre 39% de survivants.

b. Pclass : La classe du passager

Comme un avion, il y a plusieurs classes dans un bateau, la première classe est la plus prestigieuse. Voyons quel est le taux de survie par classe, sns. barplot permet de générer un diagramme en barres de y(Survived) en fonction de la valeur x (ici Pclass).

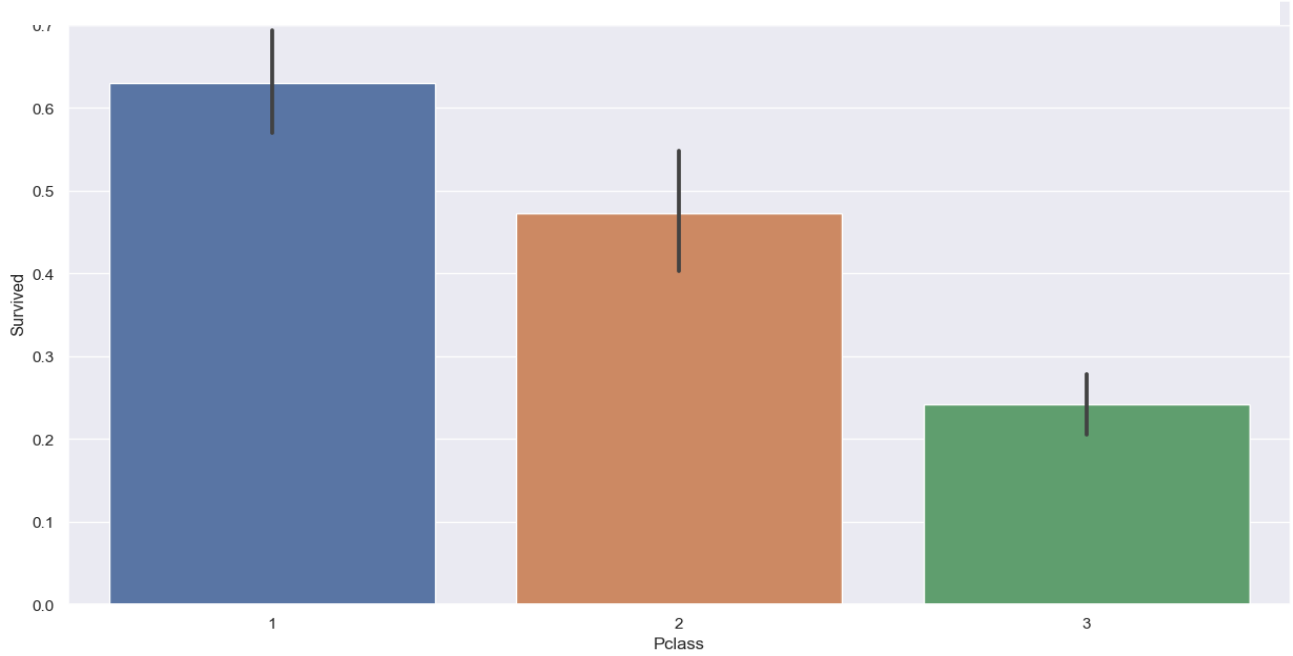
```
# liste des valeurs prises par Pclass
data.Pclass.value_counts()
```

```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

On peut voir que Pclass prend 3 valeurs, 1,2 ou 3, il n'y a pas de valeur aberrante ni de valeurs manquantes.

```
_ = sns.barplot(x="Pclass",y="Survived", data=data)
```

Figure 2:Repartition des classes de passagers par rapport aux survivants



Visuellement il y a une forte dépendance entre classe et survie. En effet on peut voir que ceux de la classe 1 ont tendance à survivre plus que ceux des autres classes par exemple. Voyons dès à présent qu'el est le niveau de corrélation.

Nous allons juste faire appel à notre fonction de cramer que nous avons créé pour mesurer la corrélation entre nos deux variables qualitatives.

```
print(cramers_v(data['Pclass'], data['Survived']))
```

```
0.33668387622245516
```

On peut voir que le Coefficient de Cramer est de l'ordre de 33% ce qui montre une relation de dépendance entre ces variables.

Voici également visuellement sur un tableau croisé comment se reparties les données.

```
pd.crosstab(data["Survived"],data["Pclass"])
```

Survived \ Pclass	1	2	3
0	80	97	372
1	136	87	119

Au vue dette répartition on peut allons confirmer que cette variable est pertinente donc on peut le conserver.

c. *Exploitation du nom et création de la variable Title(titre)*

Nous allons visualiser le nom et dire les raisons du choix du titre.

```
##Name
data.Name[0]
```

```
'Braund, Mr. Owen Harris'
```

Le nom (champ Name) est constitué du « nom + titre + prénom », et on peut en tirer au moins 1 informations intéressantes : Le titre.

Le choix de ce dernier s'est fait d'abord d'un point de vue subjectif et ensuite par le biais de nos critères. Du point subjectif on peut dire que :

- Le colonel ou le capitaine par exemple à plus de chance de se sauver, grâce à sa formation militaire et sa force par exemple
- De même les petits enfants ou garçons ou filles (Master) ont plus de risque de mourir vu leur age, et leur force physique et mental.
- Plus encore le révérend qui est un homme de Dieu peut par exemple accepter de se sacrifier pour sauver les autres ;

Par ailleurs, nous allons extraire cette variable et faire de traitement et voir si elle vérifie nos critères.

❖ Extraction de la variable

Nous allons extraire le titre de la variable name :

```
#Extraire le titre
data['Title'] = data['Name'].str.split(", ", expand=True)[1].str.split(".", expand=True)[0]
```

Nous allons visualiser alors cette variable :

```
data.Title.value_counts()
```

```
Mr              517
Miss            182
Mrs             125
Master           40
Dr                7
Rev               6
Mlle              2
Major             2
Col               2
the Countess      1
Capt              1
Ms                1
Sir               1
Lady              1
Mme               1
Don               1
Jonkheer          1
Name: Title, dtype: int64
```

- Critère du choix de la variable Title

Nous allons croiser cette variable avec la variable survived :

```
pd.crosstab(data.Title,data.Sex)
```

Sex	female	male
Title		
Capt	0	1
Col	0	2
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1
the Countess	1	0

Il y a beaucoup de titres peu fréquents, on va fusionner les titres pour lesquels on n'a peu de valeurs, ainsi on les regroupera en 4 catégories : Mr, Mrs, Miss, Master.

```
#Catégorisation de noms
data['Title']=data['Title'].replace(['Major','Don','Jonkheer','Capt','Sir','Rev','Col'], 'Mr')
data['Title']=data['Title'].replace(['Mlle'], 'Miss')
data['Title']=data['Title'].replace(['Lady','the Countess','Mme','Ms','Dona'], 'Mrs')
#repartition selon le sexe
data.loc[(data.Sex== "female") &(data.Title=="Dr"),'Title']="Mrs"
data.loc[(data.Sex== "male") &(data.Title=="Dr"),'Title']="Mr"
data.Title.value_counts()
```

```
Mr        537
Miss      184
Mrs       130
Master     40
Name: Title, dtype: int64
```

On va évaluer le coefficient de corrélation entre les variables .

```
print(cramers_v(data['Title'], data['Survived']))
```

```
0.5711100322192953
```

On peut voir que le coefficient de Cramer est de 0,57 qui est supérieur à 0,5 donc il existe une forte dépendance entre la variable titre et à la variable cible. On la conserve donc.

Pour une analyse plus poussée dans la suite (arbre de décision), nous allons encoder cette variable :

```
encoder = LabelEncoder()
data['Title']= encoder.fit_transform(data.Title)
data['Title']
```

```
0      2
1      3
2      1
3      3
4      2
      ..
886    2
887    1
888    1
889    2
890    2
```

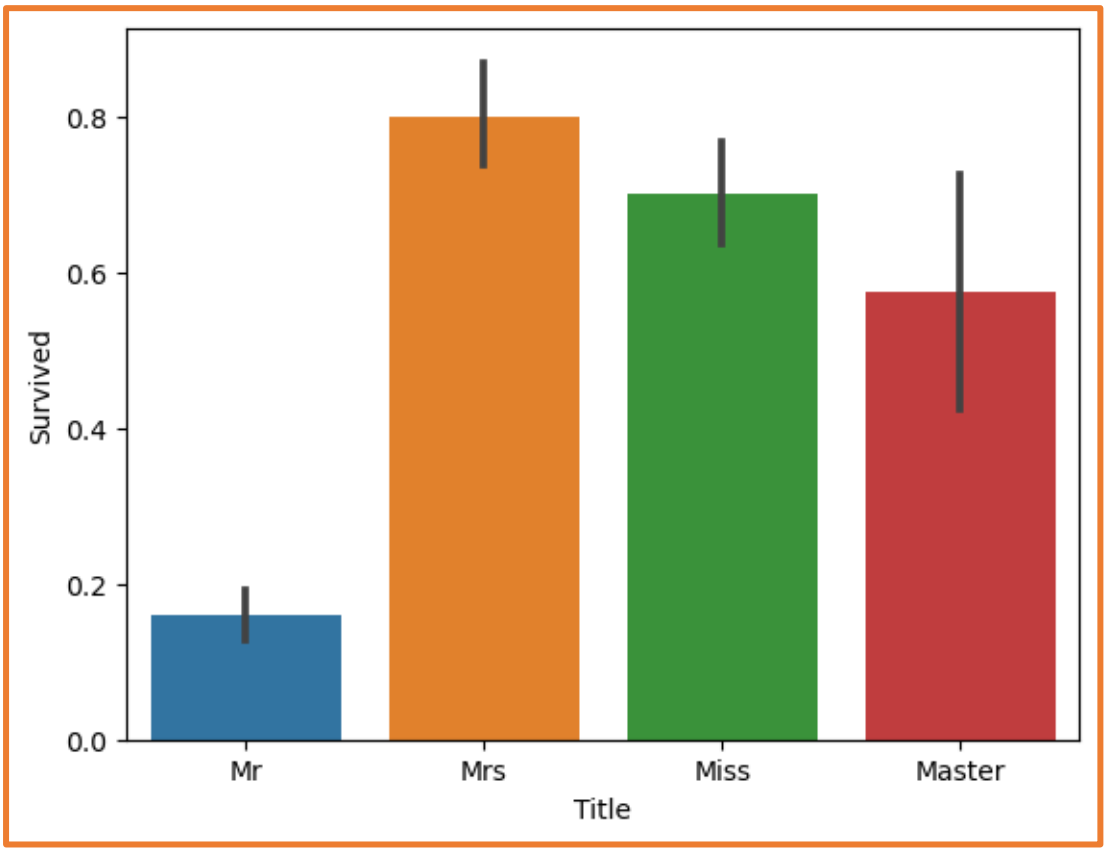
.

Ainsi on a les encodages suivants :

- 1-Miss
- 2-Mister

- 3-Mrs
- 4-Master

On peut aussi voir la répartition du titre par rapport au taux de survie :



On peut voir ici que les Mrs ont un taux de survie plus élevé que les autres.

d. Variable Sexe (Sex)

Une vue globale de cette variable nous montre qu'elle ne comporte pas de valeurs manquantes et aberrantes.

```
data.Sex.value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

Nous allons dès à présent mesurer la corrélation entre les variables

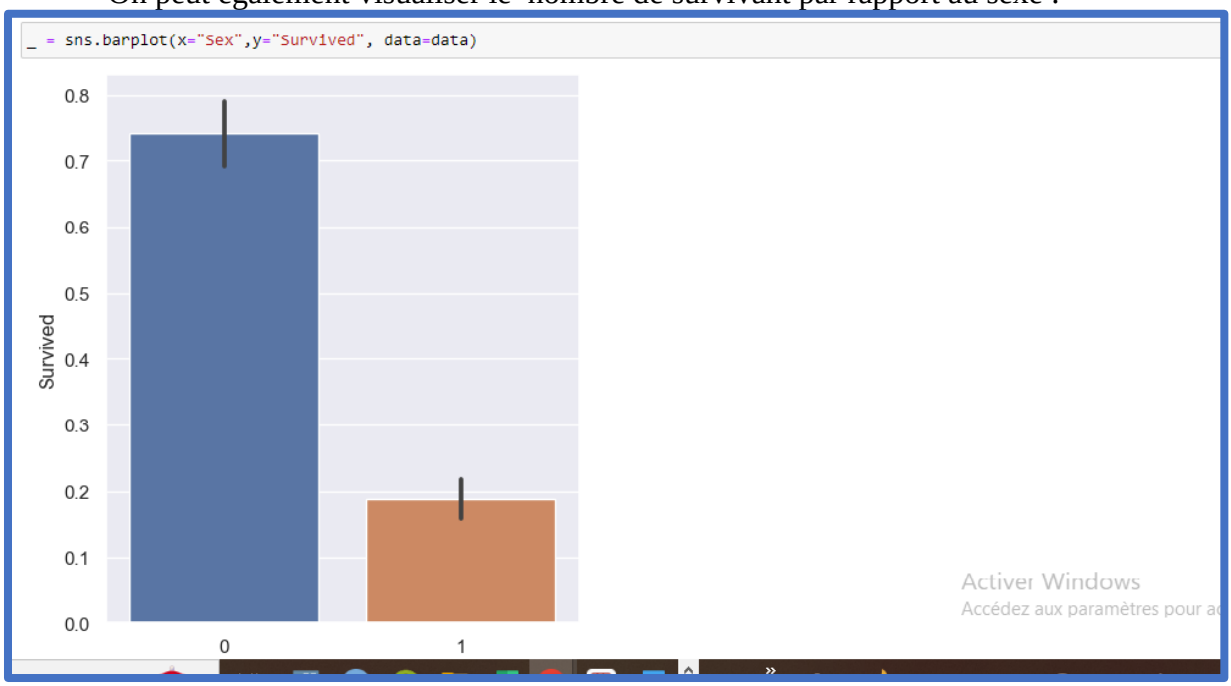
```
print(cramers_v(data['Sex'], data['Survived']))
```

```
0.5401999468101071
```

Donc le Phi de Cramer est supérieur à 0,5 donc il existe une forte dépendance entre la variable Sex et Survived donc on conserve cette variable Sex.
Pour des analyses poussées nous allons encoder la variable Sex.

```
#Sex(encoder)
encoder = LabelEncoder()
data['Sex']= encoder.fit_transform(data.Sex)
data['Sex']#Sex(encoder)
#1-Male
#0-Female
```

On peut également visualiser le nombre de survivant par rapport au sexe :



On peut voir que le taux de survie au niveau des hommes est supérieur au taux de survie au niveau des femmes.

e. Variable Age

- Valeurs manquantes

Cette variable rend compte comme le nom l'indique de l'âge des passagers.

Nous allons voir si elle ne comporte pas de valeurs manquantes.

```
data['Age'].isnull().sum()
```

```
177
```

Il manque beaucoup de données, et pas question de supprimer les passagers pour lesquels il manque l'âge. Nous allons donc procéder à une correction de ses valeurs.

- Traitement de ces valeurs manquantes

Pour traiter ces valeurs manquantes, nous pouvons opter ^pour un remplacement des valeurs manquantes par la moyenne ou la médiane mais cela pourrait biaiser beaucoup nos analyser car rien ne nous confirme que tous les âges ont les mêmes caractéristiques. Pour ce faire donc, nous avons croiser l'âge avec plusieurs variables de notre base et nous avons pris la décision de corriger cette dernière avec la variable Titre (Title).

-D'abord visualisons la répartition de cette dernière par rapport aux valeurs manquantes de l'âge dans un tableau croisé.

```
pd.crosstab(data.Age.isna(),data.Title)
```

Title	Master	Miss	Mr	Mrs
Age				
False	36	148	417	113
True	4	36	120	17

On peut voir que Master comporte 4 valeurs manquantes ; Miss : 36 valeurs manquantes ; Mr : 120 valeurs manquantes et Mrs 17 valeurs manquantes.

-Nous allons alors corriger ces valeurs en remplaçant pour chaque catégorie ; les valeurs manquantes par les moyennes correspondantes.

```
#Moyenne Mrs
data.loc[(data.Age.notna())&(data.Title=="Mrs"),'Age'].mean()
#35.92 --36 ans
```

```
35.92035398230089
```

```
#Moyenne Miss
data.loc[(data.Age.notna())&(data.Title=="Miss"),'Age'].mean()
#22 ans
```

```
21.804054054054053
```

```
#Moyenne Mr
data.loc[(data.Age.notna())&(data.Title=="Mr"),'Age'].mean()
#32.94 --33 ans
```

```
32.98441247002398
```

```
#Moyenne Mr
data.loc[(data.Age.notna())&(data.Title=="Master"),'Age'].mean()
  #4.57--5 ans
```

```
4.574166666666667
```


Ainsi nous allons remplacer ces valeurs manquantes par les moyennes des âges des catégories.

```
data.loc[(data.Age.isna())&(data.Title=="Mrs"),'Age']=36
```

```
data.loc[(data.Age.isna())&(data.Title=="Miss"),'Age']=22
```

```
data.loc[(data.Age.isna())&(data.Title=="Mr"),'Age']=33
```

```
data.loc[(data.Age.isna())&(data.Title=="Master"),'Age']= 5
```

```
data['Age'].isna().sum()
```

```
0
```

On peut voir avec le dernier code qu'il n'y a plus de valeurs manquantes on peut allors visualiser notre base et voir si on peut conserver la variable age.

Comme cette dernière présente des valeurs assez éloignées nous allons centrer et réduire cette variable ce qui nous permettra non seulement de faire la suite mais également de faire le test de cramer dessus (Car elle devient qualitative après recodage et les moyennes et variances n'ont plus de sens) :

```
data['Age']=StandardScaler().fit_transform(data['Age'].values.reshape(-1, 1))
```

- ❖ Critère du choix de la variable age

Nous allons tous simplement utiliser la valeur de la corrélation de cramer pour voir si l'on peut utiliser cette variable. Nous allons le prendre si le coefficient est supérieur à 0,15.

```
print(cramers_v(data['Age'], data['Survived']))
```

```
0.2707481148791006
```

On trouve 0,27 qui est supérieur à 0,15 donc on peut utiliser cette variable.

f. Variable SibSp et Parch

Ces variables rendent compte d'une part des sœurs, frères ('SibSp') et des parents et enfants (Parch). Ces variables ne comportent pas de valeurs manquantes comme nous le montre la ligne suivante ainsi que des valeurs aberrantes. Et elle ont un bon coefficient de corrélation de cramer.

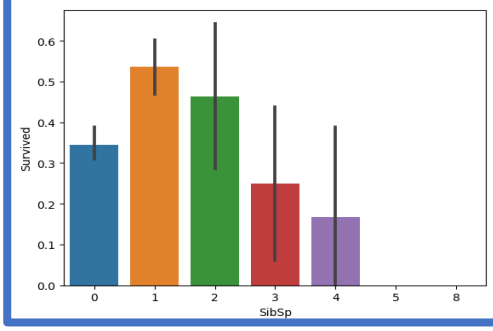
- *SibSp*

```
##SibSp
data.SibSp.value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
#Sibp
sns.barplot(x='SibSp',y='Survived',data=data)
```

```
<AxesSubplot:xlabel='SibSp', ylabel='Survived'>
```




```
print(cramers_v(data['SibSp'], data['Survived']))
```

```
0.18742816095927223
```

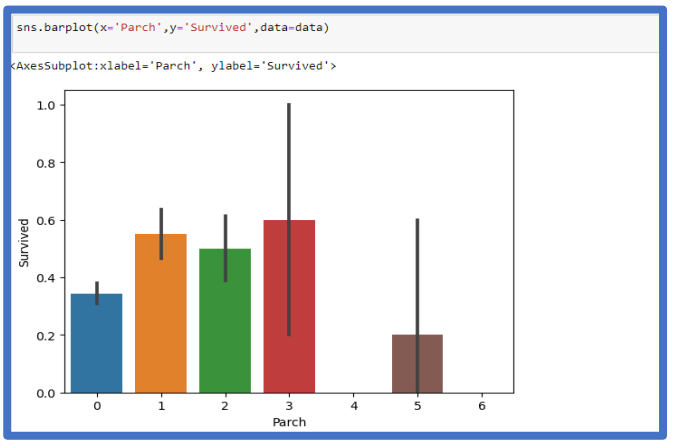
- Parch

```
###Parch
data.Parch.value_counts()
```

```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

```
sns.barplot(x='Parch',y='Survived',data=data)
```

```
<AxesSubplot:xlabel='Parch', ylabel='Survived'>
```



```
print(cramers_v(data['Parch'], data['Survived']))
```

```
0.15693364431605167
```

On peut voir en globalité que leur coefficient de Cramer est supérieur à 0,15 donc on conserve ces variables SibSp et Parch.

g. Variable Embarked

Nous allons visualiser cette variable et voir si elle est pertinente.

```
#Embarked
data.Embarked.value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

Il n'y a pas de valeurs aberrantes mais 2 valeurs manquantes.

- Traitement

Ici on a 2 valeurs manquantes. Le croisement avec la variable « Tickets » pour voir peut-être les personnes ayant les mêmes tickets pour déduire le lieu d'embarcation ; s'avère inutile car seul ces deux passagers aux valeurs manquantes ont ces tickets.

```
data[data.Embarked.isna()]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	1	Icard, Miss. Amelie	0	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	0	62.0	0	0	113572	80.0	B28	NaN

Ainsi nous allons chercher les autres personnes qui possède ce ticket :

```
pd.crosstab(data.Ticket == "113572",data.Embarked)
```

Embarked	C	Q	S
Ticket			
False	168	77	644

On a aucune information c'est à dire personne n'a ces tickets à part eux.
Donc pour éviter toute supposition qui pourrait biaiser les résultats et aussi comme les valeurs manquantes ne représentent que 0,2% ce qui est négligeable alors on a décidé alors de le supprimer :

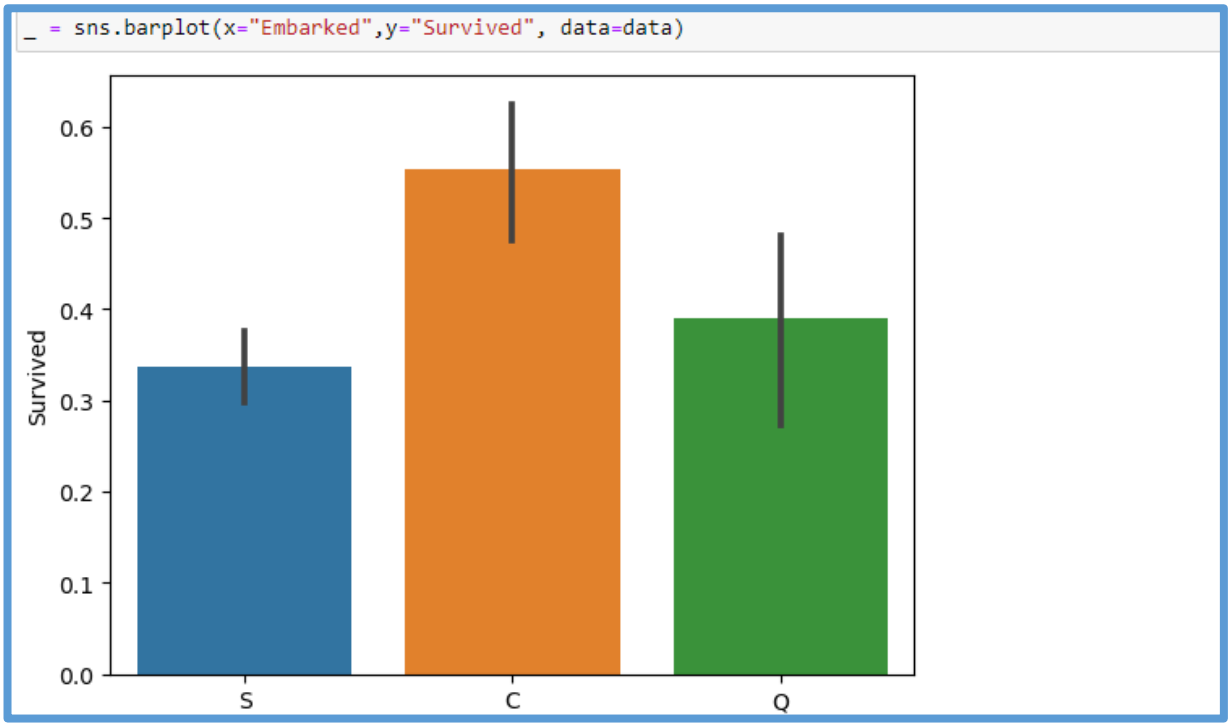
```
##Aucune informarion concernant on drop
data = data.dropna(subset=['Embarked'])

##Valeurs manquantes
data.Embarked.isna().sum()
##0
```

```
0
```

❖ Critère du choix

Nous allons voir la répartition du taux de survie par rapport au lieu d'embarcation.

```
_ = sns.barplot(x="Embarked",y="Survived", data=data)
```



On peut voir que ceux qui descendent au lieu C ont plus de chance de survivre que les autres.

A présent évaluons le coefficient de corrélation entre les variables.

```
print(cramers_v(data['Embarked'], data['Survived']))
```

```
0.16605833339661635
```

On peut alors conserver cette variable. Nous l'encoderons pour la suite du travail.

```
Encoder = LabelEncoder()
data['Embarked'] = encoder.fit_transform(data.Embarked)
print(data['Embarked'])
#2= S; 0=C ; 1=Q
```

```
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 891, dtype: int32
```

h. Récapitulatif

Nous venons de traiter nos variables et avons choisis les 7 variables que nous allons utiliser pour notre classification et les raisons de ce choix. Ces variables ne sont rien d'autre que :

- ✓ Titre (Title) [« extraite de la variable Name]
- ✓ Classe des passagers (Pclass)
- ✓ Age des passagers (Age)
- ✓ Le nombre de frères et Sœurs à bord (SibSp)
- ✓ Le nombre de Parents et enfants à bord (Parch)
- ✓ Le Sexe (Sex)
- ✓ Le lieu d'embarcation (Embarked)

La question à se poser maintenant est : Pourquoi choisir ces variables et non les autres ? Nous allons donner les raisons dans cette dernière partie du traitement.

i. Raison du non choix des autres variables

Nous allons prendre chaque variable restante et donner les raisons qui seront liés à nos opinions, nos expériences et d'autre même aux chiffres.

- *Nom (Name) et Identifiant des passagers (Passenger Id)*

Ici on a choisi que titre mais on a laissé les noms et prénoms. Pourquoi ?

La raison est que l'on veut prédire la suivie et ce n'est pas parce qu'on t'appelle X ou Y que tu vas survivre. Le nom est indépendant de la survie. Il en est de même pour l'identifiant des passagers

- *Prix du billet (Fare)*

Ici on a laissé cette variable non parce qu'elle n'apporte pas d'information mais tout simplement pour des raisons de multi colinéarité et de répétition. En effet connaissant le lieu d'embarcation d'un passager et sa classe on peut facilement déduire le prix de son ticket.

- *Tickets (Ticket)*

Ici simplement, avec le lieu d'embarcation et la classe on peut connaitre ton ticket. C'est le processus qui est peut-être utilisé pour accorder des tickets à la gare

- Cabine (Cabin)

Il y a trop peu de données sur la cabine, par ailleurs la classe du passager définit l'emplacement de sa cabine, pour ne pas générer trop de bruit nous allons laisser cette variable.

Nous allons maintenant implémenter notre modèle de classification.

III. IMPLÉMENTATION DU MODÈLE DE CLASSIFICATION AVEC LES ARBRES DE DÉCISIONS : MÉTHODE DU HOLDOUT VALIDATION

1. *Extraction de la variable cible et les descripteurs*

La variable cible est « Survived » et les autres variables choisis se regroupent au niveau des descripteurs.

```
##Choix de target
Y=data['Survived']
```

```
X=data[['Pclass','Sex','SibSp','Parch','Embarked','Title','Age']]
```

2. *Méthode du Holdout validation*

Avant de construire notre modèle, nous allons séparer nos données en deux parties : 70% pour l'apprentissage et 30% pour la prédiction. C'est la première étape de holdout_validation

```
#Holdout validation_etape1
#Separation de nos données en deux parties 70% pour l'apprentissage et 30% pour le test
Xtrain,Xtest,ytrain,ytest=train_test_split(X,Y,random_state=0,train_size=0.7)
```

Nous pouvons maintenant construire un arbre de décisions sur ces données.

```
#appel de la fonction
model=DecisionTreeClassifier()
#Entrainer le modele
result = model.fit(X,Y)
```

Ainsi nous venons d'implémenter le modèle par la méthode du holdout validation.

IV. GRAPHIQUE DU MODELE ET AVANTAGE

1. *Graphique du modèle*

```
Y=str(Y)
```

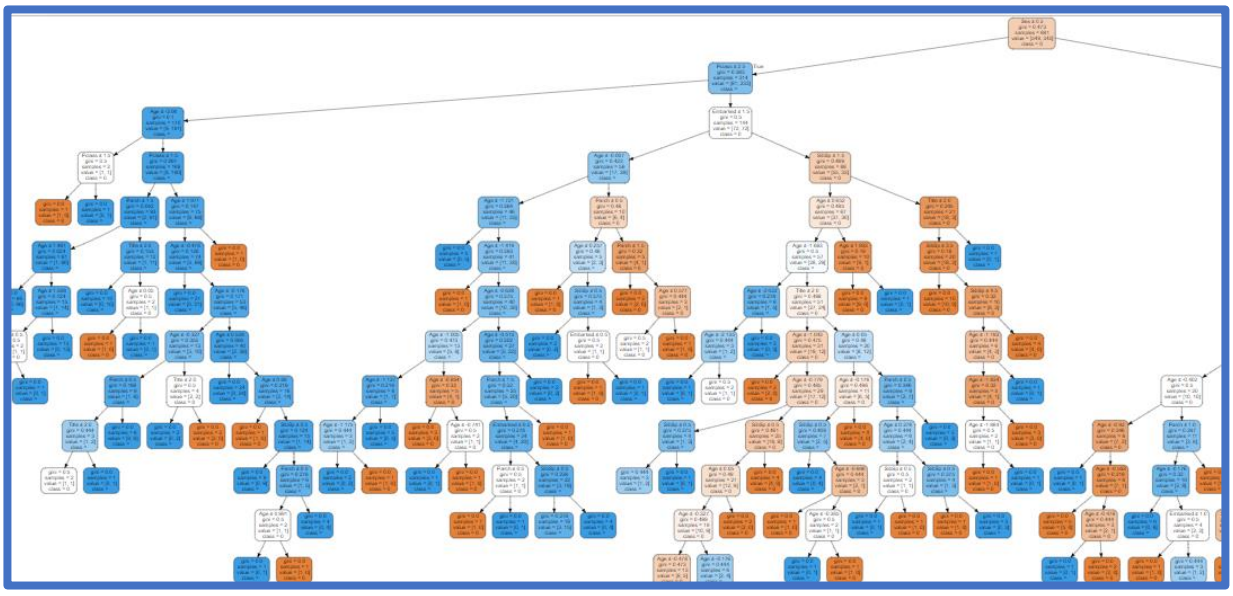
```
graph = export_graphviz(model,
                                feature_names = X.columns,
                                out_file = None,
                                class_names = Y,
                                special_characters = True,
                                filled = True,
                                rounded = True
                                )
```

Visualisons l'arbre de décision

```
graph2 = graphviz.Source(graph)
```

```
graph2
```

Arbre de décision :



Extrait de l'arbre de décision. Nous enverrons le fichier PDF de l'arbre de décision accompagné du Word

2. *Avantage du graphique*

Parmi les avantages des arbres de décision :

- ✓ Ils sont simples à comprendre et à interpréter. On peut visualiser les arbres. Aussi, on peut expliquer les résultats obtenus facilement.
- ✓ Ils peuvent travailler sur des données avec peu de préparation. Par exemple, ils n'ont pas besoin de la normalisation des données.
- ✓ Ils acceptent les données numériques et nominales. Les autres algorithmes d'apprentissage sont spécialisés dans un seul type de données.
- ✓ Ils donnent de bonne performance même si leurs hypothèses sont un peu violées par le modèle réel à partir duquel les données ont été générées.
- ✓ Transparence : un arbre de décision permet à votre équipe et vous-même de prendre des décisions à l'aide d'une approche ciblée, ce qui est sans doute le principal avantage de cet outil. Analyser chacune de vos décisions et en calculer la valeur attendue, vous permettra de déterminer clairement celle à privilégier.
- ✓ Efficacité : créer un arbre décisionnel est très rapide et vous n'avez pas besoin de beaucoup de ressources. Cet outil s'avère donc particulièrement efficace. En revanche, d'autres outils de prise de décision (enquêtes, ou prototypes) s'étalent parfois sur plusieurs mois et sont particulièrement onéreux. Un arbre de décision est un moyen simple et efficace pour décider des mesures à prendre.
- ✓ Flexibilité : si vous trouvez une nouvelle idée alors que vous avez commencé votre arbre, vous pouvez ajouter cette dernière très facilement. En outre, si vous obtenez de nouvelles données au cours de votre analyse, vous pouvez aussi ajouter des branches pour schématiser les nouveaux résultats possibles

V. PERFOMANCE DU MODELE

Une fois notre modèle généré, nous devons nous assurer si elle est bonne ou pas.
Nous allons d'une part évaluer la performance de notre modèle simple et la performance dans la prédication

1. *Performance simple :*

On l'obtient par le code suivant :

```
#Perfommance du modele
result.score(X, Y)
```

```
0.937007874015748
```

On peut dire que notre modèle est bon à 93%

2. *Performance de notre apprentissage et de la prédiction*

Nous allons voir si on peut avoir confiance à notre modèle en termes de prédication.

```
y_pred_train= model.predict(Xtrain)
```

```
print(accuracy_score(ytrain,y_pred_train))
```

```
0.9437299035369775
```

```
y_pred_test=model.predict(Xtest)
```

```
print(accuracy_score(ytest,y_pred_test))
#0.98
```

```
0.9213483146067416
```

Sur base d'apprentissage comme la base test, notre score est supérieur à 90% ce qui dénote d'une bonne prédication de la part de notre modèle.

PARTIE 2 : RECHERCHE

INTRODUCTION

Les algorithmes de clustering identifient les sous-groupes intrinsèques dans un ensemble de données ou dans un ensemble des éléments qui montrent des similitudes par paires plus élevées par rapport aux autres sous-groupes. Bien que leur utilisation soit relativement répandue, le manque d'informations a priori complique l'évaluation de solutions de clustering. Les tentatives pour relever ce défi sont généralement s'appuyer sur la mise en place de démarches de validation interne, qui se concentrent sur les quantités et les fonctionnalités inhérentes à une solution de regroupement. Mais cette dernière s'avère insuffisante voire limiter pour confirmer la qualité de nos classifications. Pour faire face a ces situations une catégorie d'indices, complexes, ont été créés ; il s'agit des indices relatifs. Ce sont ces indices qui feront l'objet de notre partie. Nous allons axer notre travail sur 4 points essentiels à savoir :

-Le fonctionnement des indices relatifs
-Les élements généraux de comparaison avec les autres indices
-La liste de quelques indices relatifs
-La description d'un indice relatif en détail

I. DESCRIPTION DU FONCTIONNEMENT DES INDICES RELATIFS

Alors que les indices internes évaluent une seule réalisation de l'algorithme de clustering, ils ne peuvent pas détecter la stabilité de l'algorithme face aux variations des données, ou la cohérence des résultats en cas de redondance. Une famille d'indices plus complexes, appelés indices de validation relative, tente de mesurer la cohérence d'un algorithme en comparant les clusters obtenus par le même algorithme dans des conditions différentes. Ces indices tentent souvent d'exploiter la redondance des données.

- ❖ Comment fonctionne alors ces indices ?

Ces indices même s'ils n'ont pas les mêmes critères, ils suivent la même logique. On peut regrouper en deux grandes étapes :

- ✓ Le partitionnement ou regroupement des données en Cluster selon des critères bien définis

  Par exemple certains indices ont déjà le nombre de clusters définis au début alors que d'autre se base sur certaines règles pour partitionner. C'est le cas de certains indices qui se base sur une caractéristique donnée pour faire le regroupement. Parlant de critère, il faut savoir que le partitionnement ne se fait pas juste comme ça, elle se base sur certaines règles. Certains indices par exemple, ils sont calculés en regroupant les échantillons après avoir supprimé la caractéristique donnée et en mesurant la taille de ces grappes (c'est-à-dire la distance moyenne entre tous les échantillons et les centroïdes de leur grappe) spécifiquement sur la caractéristique inspectée.

- ✓ Le cluster obtenu après partionnement est comparé avec d'autre Clusters de référence ou doit vérifier certaines conditions pour être considéré comme bon.

Par exemple pour certains indices ; les échantillons ou clusters obtenus doivent être compact ou soit après certains nombres d'itération d'un algorithme ou du partitionnement on doit retrouver les mêmes nombres de clusters avec les mêmes éléments. Une fois cette dernier étape conforme alors on peut valider notre modèle.

II. ELEMENTS GENERAUX DE COMPARAISON AVEC LES AUTRES INDICES

Plusieurs éléments permettent de comparer les indices relatifs par rapport aux autres indices :

- Le nombre de Cluster ou grappes
  - ✓ Contrairement à la validation interne, les méthodes de validation relative ont le potentiel de transformer l'analyse de cluster en un problème de sélection de modèle et d'aider à évaluer la meilleure solution de clustering (c'est-à-dire, meilleur nombre de grappes)

- Partitionnement des données.
  - ✓ La façon dont ces méthodes sont conçues offre également la possibilité de déterminer dans quelle mesure une solution de clustering se généralise à des données invisibles et donc de permettre la réplication de la partition de données choisie.

- Les logiciels
  - ✓ Bien qu'une variété de logiciels les packages contiennent des méthodes de validation de cluster internes et externes, des logiciels open source pour mettre en œuvre facilement le plein potentiel des techniques de validation relative font défaut.

- Erreur de regroupement
  - ✓ En ce qui concerne les erreurs de regroupement, les indices internes et relatifs peuvent présenter un comportement similaire. Cependant, dans le cas de modèles complexes et de clusters, une approche basée sur la minimisation de l'erreur de prédiction peut être particulièrement avantageux car les indices internes ont tendance à échouer bien corréler avec les erreurs

- Cout des calculs
  - ✓ Les indices internes et externes sont moins coûteux en calcul en terme de correction d'erreurs de regroupement même si elles ne sont pas efficaces que les indices relatifs

On peut voir que plusieurs éléments généraux sont à prendre en compte dans cette comparaison .

III. LISTE DE QUELQUES INDICES RELATIFS

Nous allons des à présents citer quelques indices relatifs. De part nos recherches nous avons eu à rencontrer certains indices qui sont :

- Indices de Merit (FOM: Figure of merit)

Elle suppose que la redondance est intégrée dans les données de l'échantillon pour mesurer la cohérence. Le FOM d'une caractéristique est calculé en regroupant les échantillons après avoir supprimé la caractéristique donnée et en mesurant la taille de ces grappes (c'est-à-dire la distance moyenne entre tous les échantillons et les centroïdes de leur grappe) spécifiquement sur la caractéristique inspectée. Si cette distance moyenne est petite, alors l'algorithme de regroupement a partitionné les échantillons en grappes compactes même avec la caractéristique supprimée, ce qui suggère que l'algorithme est cohérent. Le FOM global pour un algorithme de clustering est la somme de ces valeurs sur toutes les fonctionnalités, en laissant chacune de côté une à la fois. D'un point de vue heuristique, un algorithme de clustering qui produit des clusters cohérents devrait être capable de prédire les fonctionnalités supprimées et aurait donc un indice FOM faible.

- La stabilité des Clusters
- Indice de Dunn
- Indice de Huberts
- Indice de Davies et Bouldin

IV. LA STABILITE DES CLUSTERS : PRINCIPES ET MESURES

1. *Principes*

L'indice d'instabilité mesure la capacité d'un ensemble de données groupées à prédire le regroupement d'un autre ensemble de données échantillonné à partir de la même source.

La stabilité est mesurée en divisant d'abord l'ensemble de points à regrouper en deux parties. L'algorithme de clustering à l'étude est appliqué à la première partie, et les étiquettes obtenues sur ces points sont utilisées pour former un classifieur qui partitionne tout l'espace. L'algorithme de regroupement d'origine et ce classificateur sont appliqués à la deuxième collection de points, générant deux ensembles d'étiquettes. Le désaccord entre ces étiquettes, moyenné sur des courses répétées

Le désaccord entre ces étiquettes, moyenné sur des partitions aléatoires répétées des points, définit l'instabilité de l'algorithme de clustering.

L'indice d'instabilité dépend du nombre de clusters et doit donc être normalisé lorsqu'il est utilisé pour la sélection du modèle. La normalisation est obtenue en divisant par l'instabilité obtenue en utilisant un estimateur aléatoire comme classificateur.

Un autre problème affectant l'indice d'instabilité est la sélection de la règle de classification, qui peut fortement influencer les résultats. De plus, des études de simulation montrent que cet indice ne fonctionne pas mieux que d'autres indices relatifs et internes, bien qu'il soit l'un des indices les plus chronophages car il implique l'application répétée d'un algorithme de clustering et la formation d'un classificateur.

2. *Explication simple*

Ce critère est particulièrement pertinent pour choisir le nombre de clusters : si le nombre de clusters choisi correspond à la structure naturelle des données, le clustering sera plus stable que si ce n'est pas le cas. Sur l'image ci-dessous, un algorithme qui cherche à déterminer 3 clusters va raisonnablement retrouver les trois groupes que l'on voit. Mais si on lui demande de déterminer 4 clusters, la répartition dans ces 4 clusters sera plus aléatoire et ne sera pas nécessairement deux fois la même. C'est une façon de déterminer que 3 est un meilleur nombre de clusters que 4.

CONCLUSION

On peut voir de cette partie que les indices relatifs sont d'une utilité capitale et des études doivent être approfondis pour en découvrir d'autre pour faciliter la tâche et à être plus sûr de nos clusters. Par ailleurs ; la première partie nous montre au-delà de l'étude théorique l'importance des Data Mining dans la planification.