This project is done by:

Ismail Ouahbi

Hamza Khalid

Introduction

---

1. problème

Le 15 avril 1912, au cours de son voyage inaugural, le Titanic a coulé après avoir heurté un iceberg, tuant 1502 passagers et membres d'équipage sur 2224. Taux de survie traduit de 32 %. L'une des raisons pour lesquelles le naufrage a causé de telles pertes de vie, c'est qu'il n'y avait pas suffisamment d'embarcations de sauvetage pour les passagers et l'équipage. Bien qu'il y ait eu un certain facteur de chance dans la survie au naufrage, certains groupes de personnes étaient plus susceptibles de survivre que d'autres, comme les femmes, les enfants et la classe supérieure. Ainsi, le voyage du Titanic a généré beaucoup de données que les statisticiens ont collecté à fin de les analyser et les mettre en valeur pour prédire d'autres situations et éviter la répétition d'un tel problème .

---

2. Le data set

Le data set est composé de 12 colonnes et 418 lignes où 7 colonnes sont quantitatives et 5 qualitatives, il a comme valeurs nulles 414 répartis entre plusieurs colonnes

Une brève description de chaque colonne est la suivante :

· PassengerId - ID unique d'identification du passager · Survived - Drapeau de survie (0 = mort, 1 = survie) · Pclass - Classe de billet · Name - Nom du passager · Sex - Genre (homme = homme, femme = femme) · Age - Age · SibSp - Nombre de frères et sœurs / conjoints à bord du Titanic · Parch - Nombre de parents / enfants sur le Titanic · Ticket - Numéro de ticket · Fare - tarif · Cabin - Numéro de chambre · Embarked - Point de départ (port sur Titanic)

Nous allons également donner une brève description de chaque variable. pclass = classe de billet 1 = classe supérieure (riche) 2 = classe intermédiaire (classe générale) 3 = classe inférieure (classe ouvrière)

Embarked = La définition de chaque variable est la suivante C = Cherbourg Q = Queenstown S = Southampton NaN = représente une perte de données.

---

3. Hypothèses et questions

Je me souviens à peine quand j'ai regardé le film Titanic pour la première fois, mais Titanic reste toujours un sujet de discussion dans les domaines les plus divers. Ainsi plusieurs questions se pose à ce points :

- Quel genre de peuple survit ?
- Quels sont les facteurs influençant la survie des personnes ?
- L'age des personnes survis ?
- Quelle classe dominant les classes des survivants ?
- Peut on dire que les femmes et les enfants ont une forte chance à survivre ?

##I. Exploration des données

```
#importer les packages
import pandas as pd
import numpy as np
import seaborn as sns

#charger le dataset
data = pd.read_csv('tested.csv')

#les types de données
data.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

```
#prévoir le format de donnée
data.shape
```

```
(418, 12)
```

```
#lire les 5 premières lignes
data.head()
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
```

```
4          896         1       3

                                                Name     Sex   Age  SibSp
Parch  \
0                                Kelly, Mr. James    male  34.5      0
0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1
0
2                     Myles, Mr. Thomas Francis    male  62.0      0
0
3                                Wirz, Mr. Albert    male  27.0      0
0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1
1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

```
# statistiques sur les données
data.describe()
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   418.000000  418.000000  418.000000  332.000000  418.000000
mean   1100.500000    0.363636    2.265550   30.272590    0.447368
std     120.810458    0.481622    0.841838   14.181209    0.896760
min     892.000000    0.000000    1.000000    0.170000    0.000000
25%     996.250000    0.000000    1.000000   21.000000    0.000000
50%    1100.500000    0.000000    3.000000   27.000000    0.000000
75%    1204.750000    1.000000    3.000000   39.000000    1.000000
max    1309.000000    1.000000    3.000000   76.000000    8.000000

            Parch        Fare
count  418.000000  417.000000
mean     0.392344   35.627188
std      0.981429   55.907576
min      0.000000    0.000000
25%      0.000000    7.895800
50%      0.000000   14.454200
75%      0.000000   31.500000
max      9.000000  512.329200
```

##II. Pre-traitement des données

```
#voir le nombre de valeurs nulles pour chaque colonne
data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

=> On peut constater que les trois colonnes `Age`,`Fare` et `Cabin` ont des valeurs nulles

```
#voir le nombre total de valeurs nulles
data.isnull().sum().sum()
```

```
414
```

=> Le nombre total des valeurs nulles est de 414

- **On va procéder au traitement des données manquantes :**

```
#Pour la colonne 'Fare'
data[data['Fare'].isnull()]
```

```
     PassengerId  Survived  Pclass                Name   Sex   Age
SibSp  \
152         1044         0       3  Storey, Mr. Thomas  male  60.5
0

     Parch Ticket  Fare Cabin Embarked
152      0   3701   NaN   NaN        S
```

=> Comme ce passager est de classe 3 on va remplacer la valeur manquante de colonne `Fare` par la `moyenne` des valeurs `Fare` des personnes de 3ème classe

```
#la `moyenne` des valeurs `Fare` des personnes de 3ème classe
avg_fare_p3 = np.mean(data[data['Pclass'] == 3]['Fare'])
data['Fare'].fillna(avg_fare_p3 , inplace=True)

#Vérification
data.loc[data['Fare'].isnull()]
```

```
Empty DataFrame
Columns: [PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch,
Ticket, Fare, Cabin, Embarked]
Index: []
```

```
#Pour la colonne 'Age'
data[data['Age'].isnull()]
```

```
     PassengerId  Survived  Pclass  \
10           902         0       3
22           914         1       1
29           921         0       3
33           925         1       3
36           928         1       3
..           ...       ...     ...
408         1300         1       3
410         1302         1       3
413         1305         0       3
416         1308         0       3
417         1309         0       3

                                                    Name     Sex  Age
SibSp  \
10                                     Ilieff, Mr. Ylio    male  NaN
0
22                Flegenheim, Mrs. Alfred (Antoinette)  female  NaN
0
29                                     Samaan, Mr. Elias    male  NaN
2
33   Johnston, Mrs. Andrew G (Elizabeth Lily" Watson)"  female  NaN
1
36                                   Roth, Miss. Sarah A  female  NaN
0
..                                                  ...     ...  ...
...
408                    Riordan, Miss. Johanna Hannah""  female  NaN
0
410                              Naughton, Miss. Hannah  female  NaN
0
413                                  Spector, Mr. Woolf    male  NaN
0
416                                  Ware, Mr. Frederick    male  NaN
0
417                            Peter, Master. Michael J    male  NaN
1

     Parch      Ticket     Fare Cabin Embarked
10       0      349220   7.8958   NaN        S
22       0    PC 17598  31.6833   NaN        S
29       0        2662  21.6792   NaN        C
33       2  W./C. 6607  23.4500   NaN        S
36       0      342712   8.0500   NaN        S
..     ...         ...      ...   ...      ...
408      0      334915   7.7208   NaN        Q
410      0      365237   7.7500   NaN        Q
```

```
413      0   A.5. 3236   8.0500   NaN        S
416      0      359309   8.0500   NaN        S
417      1        2668  22.3583   NaN        C

[86 rows x 12 columns]
```

=> Comme la colonne `Cabin` contient `327 valeur nulle` on va prendre une copie du dataset original(pour la préserver) et on travaille avec le dataset original où cette colonne sera supprimer

```
#prendre une copie
copie_data = data.copy()
#supprimer la colonne non désirée
data.drop(['Cabin'] , axis=1 , inplace = True)

#vérifier
data.head()
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                          Name     Sex   Age  SibSp
Parch  \
0                              Kelly, Mr. James    male  34.5      0
0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1
0
2                     Myles, Mr. Thomas Francis    male  62.0      0
0
3                              Wirz, Mr. Albert    male  27.0      0
0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1
1

    Ticket     Fare Embarked
0   330911   7.8292        Q
1   363272   7.0000        S
2   240276   9.6875        Q
3   315154   8.6625        S
4  3101298  12.2875        S
```

- **Suppression des valeurs manquantes de la colonne `Age`** :

```
#l'ensemble des valeurs manquantes de la colonne age ne représente que
20%
np.round((data.isnull().sum()['Age']/data.shape[0])*100,2)
```

```
20.57
```

```
# comme l'ensemble des valeurs manquantes de la colonne age ne
représente que 20% de l'ensemble des valeurs
# on va supprimer cette tranche
data.drop(list(data[data['Age'].isnull()]['Age'].index) , axis = 0 ,
inplace=True)

# rénitialiser les indices
data.reset_index(drop=True, inplace=True)

# vérification
data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

```
# savoir le nombre de lignes répétées
data.duplicated().sum()
```

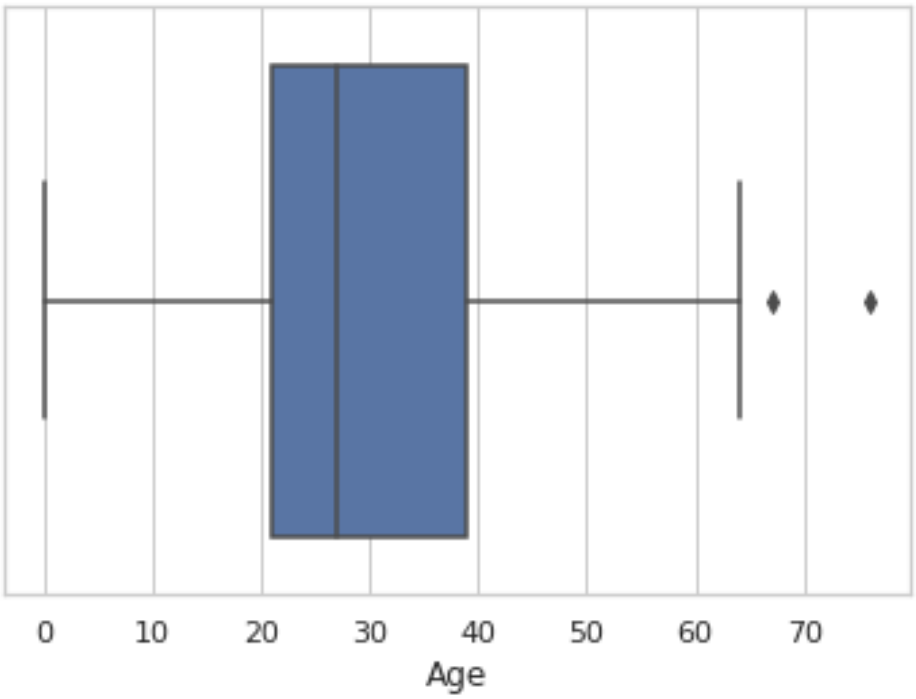
```
0
```

=> On constate que le dataset `ne contient pas de valeurs répétées`

- **Detection des outliers** :

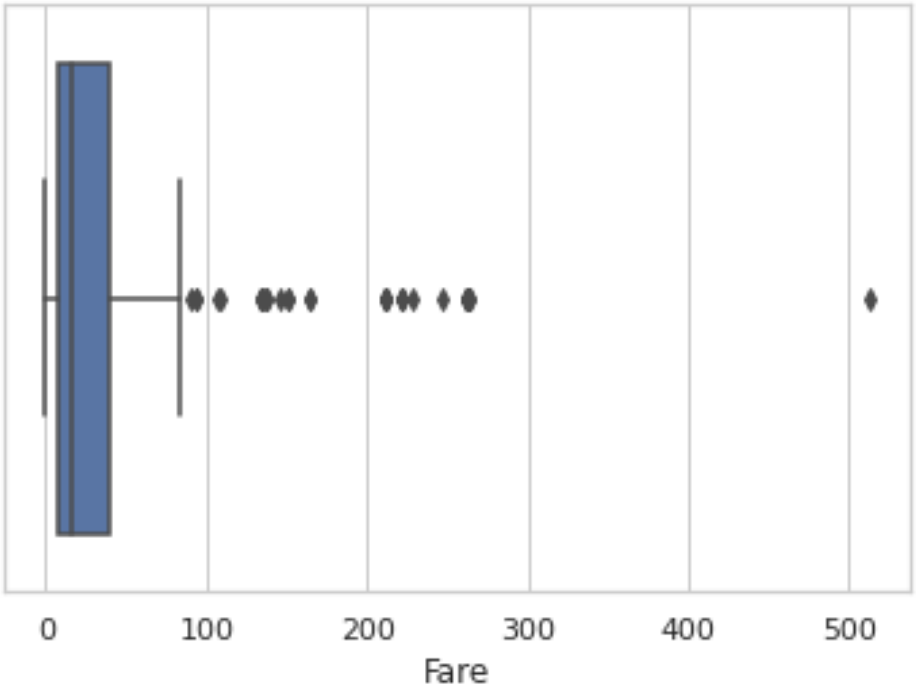
1. **Pour la variable Age** :

```
# choisir le theme seaborn
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=data['Age'])
```



1. **Pour la variable `Fare` :**

```
# choisir le theme seaborn
sns.set_theme(style="whitegrid")
ax = sns.boxplot(x=data['Fare'])
```



=> La suppression des `outliers` n'est pas toujours la solution idéale, pour notre cas les variables `Age` et `Fare` portent énormément d'informations et donc on va garder ces `outliers` .

III. Partie Analyse :

---

1. ACP :

- **Pour pouvoir appliquer `l'analyse en composantes principales`, on doit juste prendre les variables `quantitatives` :**

```
# choisir les colonnes numeriques
numerical_col = data.dtypes[data.dtypes != object].index.tolist()

# une dataframe qui ne contient que les colonnes numeriques
numerical_data = data[numerical_col]

#vérification
numerical_data.head()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	892	0	3	34.5	0	0	7.8292
1	893	1	3	47.0	1	0	7.0000
2	894	0	2	62.0	0	0	9.6875
3	895	0	3	27.0	0	0	8.6625
4	896	1	3	22.0	1	1	12.2875

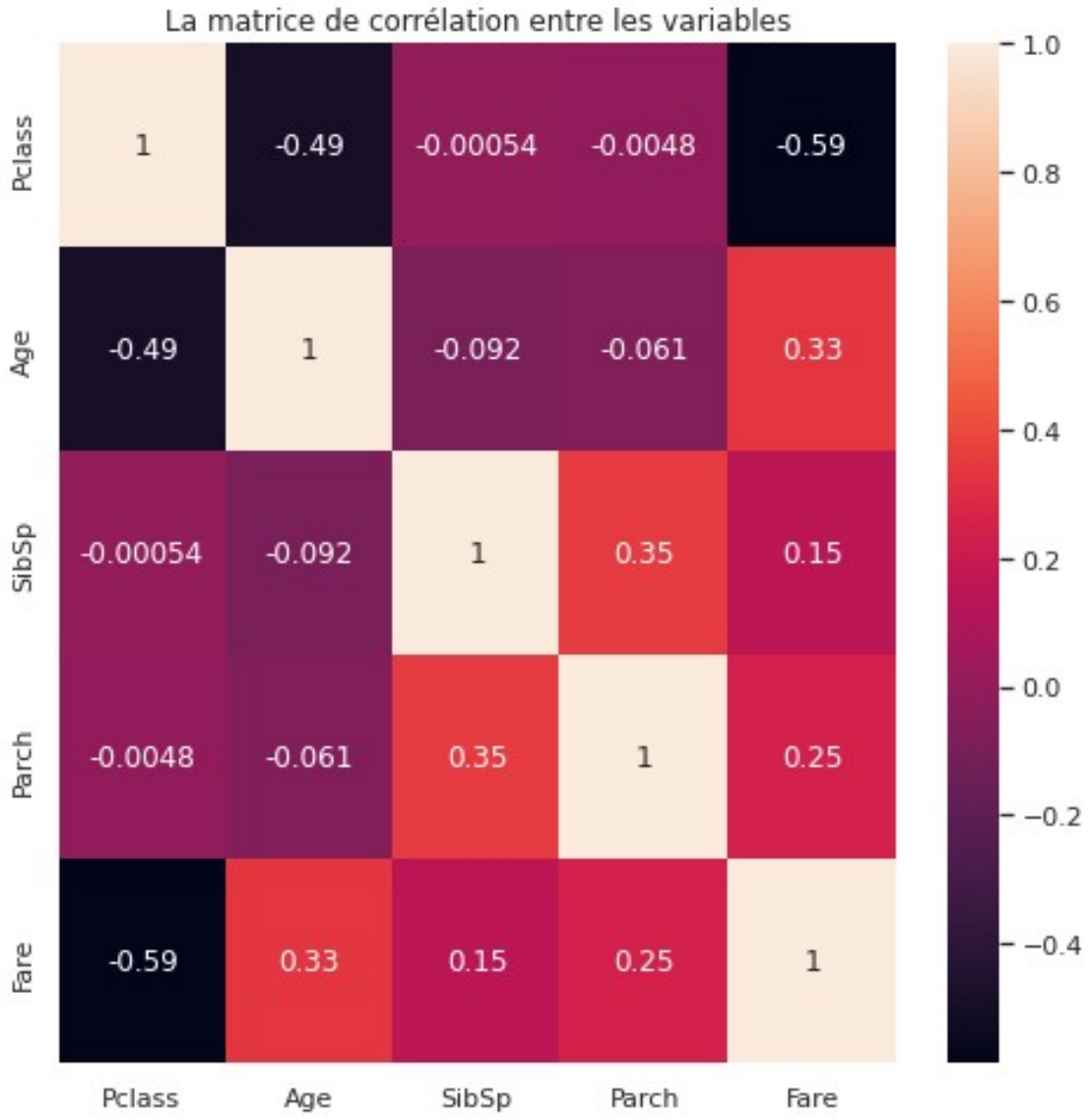
```
# suppression des colonnes numériques non utiles
numerical_data.drop(['PassengerId','Survived'], axis=1,inplace=True)
```

- **Matrice de corrélation (Heatmap)** :

```
#importer les packages nécessaires
import seaborn as sns
import matplotlib.pyplot as plt

# plot a heatmap with annotation
corr_df = numerical_data.corr(method='pearson')

plt.figure(figsize=(8, 8))
sns.heatmap(corr_df, annot=True)
plt.title('La matrice de corrélation entre les variables')
plt.show()
```



=> On constate que la corrélation entre les varibales `Age<=>Pclass` est négative
=> On constate que la corrélation entre les varibales `Pclass<=>Fare` est négative

=> Pour le reste des corrélations entre les variables, la corrélation varie entre corrélation faible et moyenne.

- **Centrer et Réduire les données :**

```
#importer les packages nécessaires
from sklearn.preprocessing import StandardScaler

# séparer les features et le target
#features
X = numerical_data
# Separating out the target
y = data[['Survived']]
# Standardizing the features
```

```
X_sc = StandardScaler().fit_transform(X)
# convertir les données en DataFrame
X_sc = pd.DataFrame(X_sc , columns = X.columns)

#vérification
X_sc
```

```
       Pclass       Age     SibSp     Parch      Fare
0    1.012325  0.298549 -0.552184 -0.491199 -0.541515
1    1.012325  1.181328  0.593598 -0.491199 -0.555094
2   -0.171097  2.240662 -0.552184 -0.491199 -0.511083
3    1.012325 -0.231118 -0.552184 -0.491199 -0.527868
4    1.012325 -0.584229  0.593598  0.744240 -0.468504
..        ...       ...       ...       ...       ...
327  1.012325 -1.926053  0.593598  0.744240 -0.444144
328 -1.354519  0.475105  0.593598 -0.491199  0.804139
329  1.012325 -0.160496 -0.552184 -0.491199 -0.542402
330 -1.354519  0.616350 -0.552184 -0.491199  1.113651
331  1.012325  0.581038 -0.552184 -0.491199 -0.551000

[332 rows x 5 columns]
```

- **Appliquer l'ACP :**

```
#importer les packages
from sklearn.decomposition import PCA
#fixer le nombre de composantes en 2
pca = PCA(n_components=2)
#ajustement
principalComponents = pca.fit_transform(X_sc)

#vérifier les données
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['composante principale 1', 'composante
principale 2'])
principalDf
```

```
     composante principale 1  composante principale 2
0                  -0.927192                -0.682985
1                  -0.380706                -0.216498
2                   0.765037                -1.487213
3                  -1.179411                -0.514033
4                  -0.998422                 1.161841
..                       ...                      ...
327                -1.643432                 1.588449
328                 1.517669                -0.198222
329                -1.153321                -0.538310
330                 1.649943                -0.946975
331                -0.793984                -0.773397

[332 rows x 2 columns]
```

```
#Ajout de la variable target
finalDf = pd.concat([principalDf, y], axis = 1)

#verification
finalDf
```

```
     composante principale 1  composante principale 2  Survived
0                  -0.927192                -0.682985         0
1                  -0.380706                -0.216498         1
2                   0.765037                -1.487213         0
3                  -1.179411                -0.514033         0
4                  -0.998422                 1.161841         1
..                       ...                      ...       ...
327                -1.643432                 1.588449         1
328                 1.517669                -0.198222         1
329                -1.153321                -0.538310         1
330                 1.649943                -0.946975         1
331                -0.793984                -0.773397         0

[332 rows x 3 columns]
```

```
# Le pourcentage d'informations garantie par chaque composante
pca.explained_variance_ratio_
```

```
array([0.39410368, 0.28590355])
```

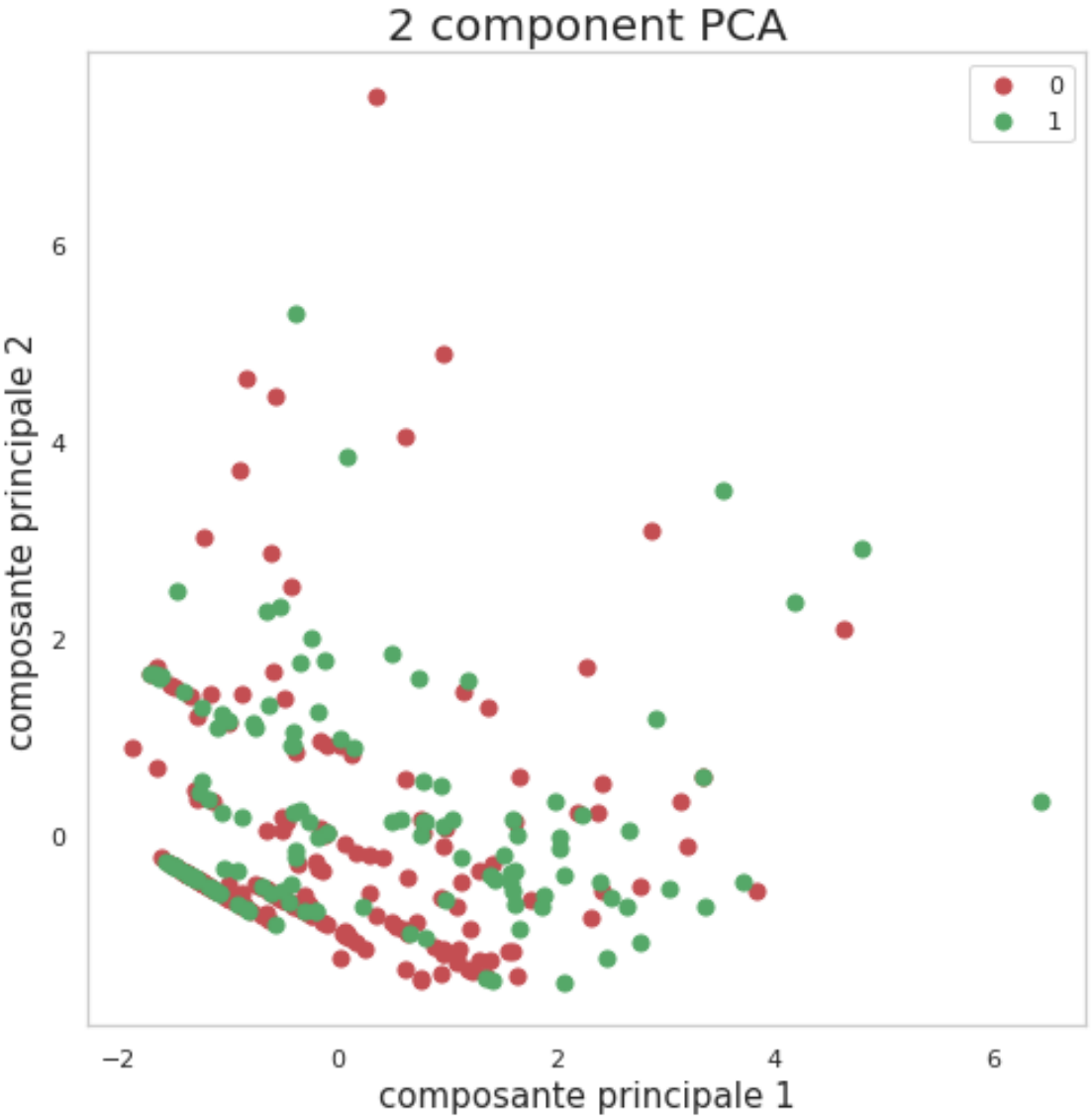
=> On constate que la première composante contient 39.41% d'informations tandis que la deuxième a 28.59% d'informations ce qui donne 68% d'informations dans le nouveau espace

- ***La visualisation des componsantes (ACP) :***

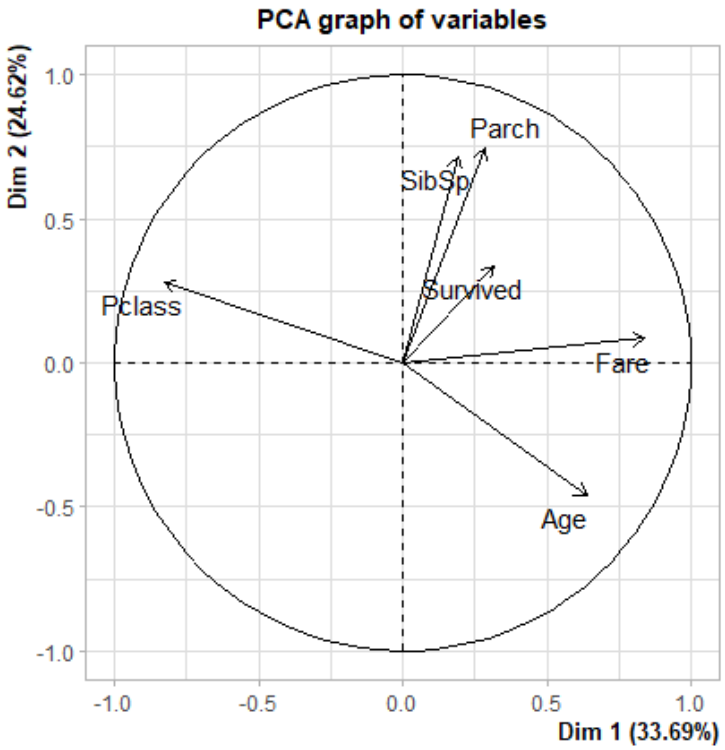
1. **Avec Python** :

```
# ajuster les dimensions de la fenetre
fig = plt.figure(figsize = (8,8))
# ajouter les sous-plot
ax = fig.subplots(1,1)
# Labeliser les axes
ax.set_xlabel('composante principale 1', fontsize = 15)
ax.set_ylabel('composante principale 2', fontsize = 15)
# ajout du titre
ax.set_title('2 component PCA', fontsize = 20)
# la variable contenant les valeurs du target
targets = [0,1]
# couleurs des plots
colors = ['r', 'g']
# processus
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['Survived'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'composante principale 1']
               , finalDf.loc[indicesToKeep, 'composante principale 2']
               , c = color
```

```
                , s = 50)
ax.legend(targets)
ax.grid()
```



1. **Avec R** :



=> On constate que la corrélation entre les varibales `Age<=>Pclass` est négative

=> La corrélation entre les varibales `SibSp<=>Parch<=>Survived` est forte

=> Les varibales `Age<=>Parch` sont indépendantes

=> Les varibales `Survived<=>Age` sont indépendantes

=> Les varibales `Survived<=>Pclass` sont indépendantes

2. AFC :

```
# on va prendre une copie du dataset pour réaliser l'AFC
afc_data = data[['Sex', 'Embarked', 'Pclass', 'Age','Survived']]

# vérification
afc_data.head()
```

```
      Sex Embarked  Pclass   Age  Survived
0    male        Q       3  34.5         0
1  female        S       3  47.0         1
2    male        Q       2  62.0         0
3    male        S       3  27.0         0
4  female        S       3  22.0         1
```

- **Comme la variable Age a plusieurs valeurs, on va les grouper selon des intervalles bins, pour mieux voir les données** :

```
# voir les statistiques sur la variable 'Age'
afc_data['Age'].describe().to_frame()
```

```
              Age
count  332.000000
mean    30.272590
std     14.181209
min      0.170000
25%     21.000000
50%     27.000000
75%     39.000000
max     76.000000
```

```
# dévision des valeurs de 'Age'
afc_data['Age_bin'] = pd.cut(afc_data['Age'],
                             [0, 21, 27, 39, 76])

# affichage
afc_data['Age_bin'].value_counts().sort_index()
```

```
(0, 21]     86
(21, 27]    81
(27, 39]    83
(39, 76]    82
Name: Age_bin, dtype: int64
```

- **Comme la variable `Pclass` a plusieurs valeurs(1,2,3…), on va les rendre qualitatives(first,second,third…)** :

```
# voir les modalités de la variables 'Pclass'
afc_data['Pclass'].value_counts().sort_index()
```

```
1     98
2     88
3    146
Name: Pclass, dtype: int64
```

```
# rendre la variable qualitative
afc_data['Pclass'].replace([1,2,3],['first','second','third'] ,
inplace=True)

# vérification
afc_data['Pclass'].value_counts().sort_index()
```

```
first      98
second     88
third     146
Name: Pclass, dtype: int64
```

- **Comme la variable `Survived` a deux valeurs(0,1), on va les rendre qualitatives(survived,not survived)** :

```
# voir les modalités de la variables 'Survived'
afc_data['Survived'].value_counts().sort_index()
```

```
0    205
1    127
Name: Survived, dtype: int64
```

```
# rendre la variable qualitative
afc_data['Survived'].replace([0,1],['not survived','survived'] ,
inplace=True)
```

```
# vérification
afc_data['Survived'].value_counts().sort_index()
```

```
not survived    205
survived        127
Name: Survived, dtype: int64
```

- **Appliquer l'AFC** :

```
# affichage des 2 premières lignes
afc_data.head(2)
```

```
      Sex Embarked Pclass   Age      Survived   Age_bin
0    male        Q  third  34.5  not survived  (27, 39]
1  female        S  third  47.0      survived  (39, 76]
```

```
#importer le package nécessaire
import prince
```

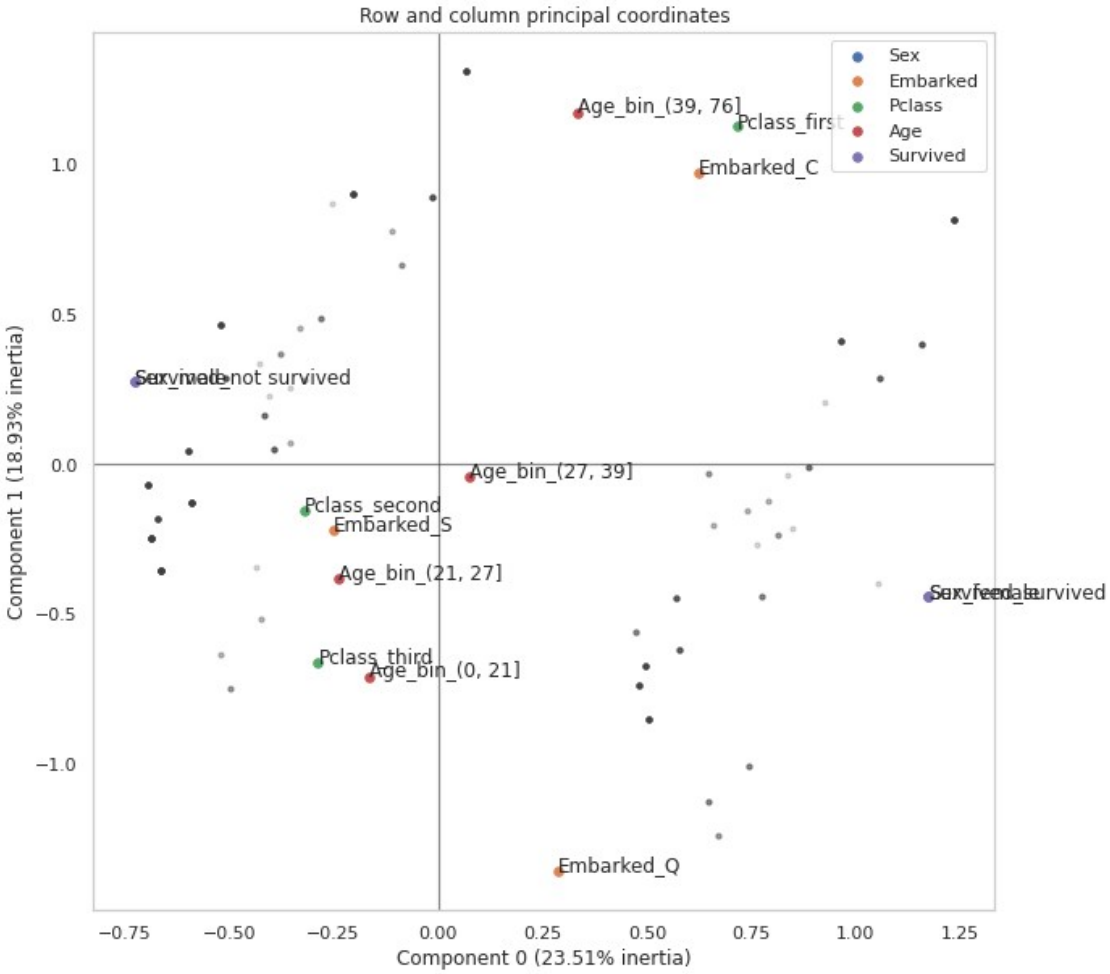
```
#prendre le dataset
afc_data = afc_data[['Sex', 'Embarked', 'Pclass',
'Age_bin','Survived']]
```

```
#instancier le model
mca = prince.MCA()
mca.fit(afc_data)
```

```
MCA()
```

```
# voir le graph
mca.plot_coordinates(afc_data,
                     row_points_alpha=.2,
                     figsize=(10, 10),
                     show_column_labels=True
                    )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f24a0b6e590>
```



=> Le genre des gens survivant est `female` car la distance entre les points qui représentent ces entités est trop petite, tandis que les gens non-survivant sont en général des `male` .

=> Les gens dont l'`age` est compris entre 39 et 76 dominent la première classe dont le point de départ est `C` .

=> Les gens dont le points de départ est `S` ont choisi la `2ème classe` .

=> On constate aussi que la variable `Age` n'a pas de corrélation avec la variable `Survived` .

3. Classification K-mean

```
# prendre une copie du dataset
data_class = data.copy()

# vérification
data_class.head()
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
```

```
3          895         0       3
4          896         1       3

                                          Name     Sex   Age  SibSp
Parch  \
0                             Kelly, Mr. James    male  34.5      0
0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1
0
2                     Myles, Mr. Thomas Francis    male  62.0      0
0
3                              Wirz, Mr. Albert    male  27.0      0
0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1
1

    Ticket     Fare Embarked
0   330911   7.8292        Q
1   363272   7.0000        S
2   240276   9.6875        Q
3   315154   8.6625        S
4  3101298  12.2875        S
```

```
# rendre la variable "Sex" quatitative
data_class['Sex'].replace(['male','female'], [1,0] , inplace=True)

# rendre la variable "Embarked" quatitative
data_class['Embarked'].replace(['S','C','Q'], [0,1,2] , inplace=True)

# filtrage et affichage
data_class = data_class[['Survived','Pclass','Sex',
'Age','SibSp','Parch','Fare','Embarked']]
data_class
```

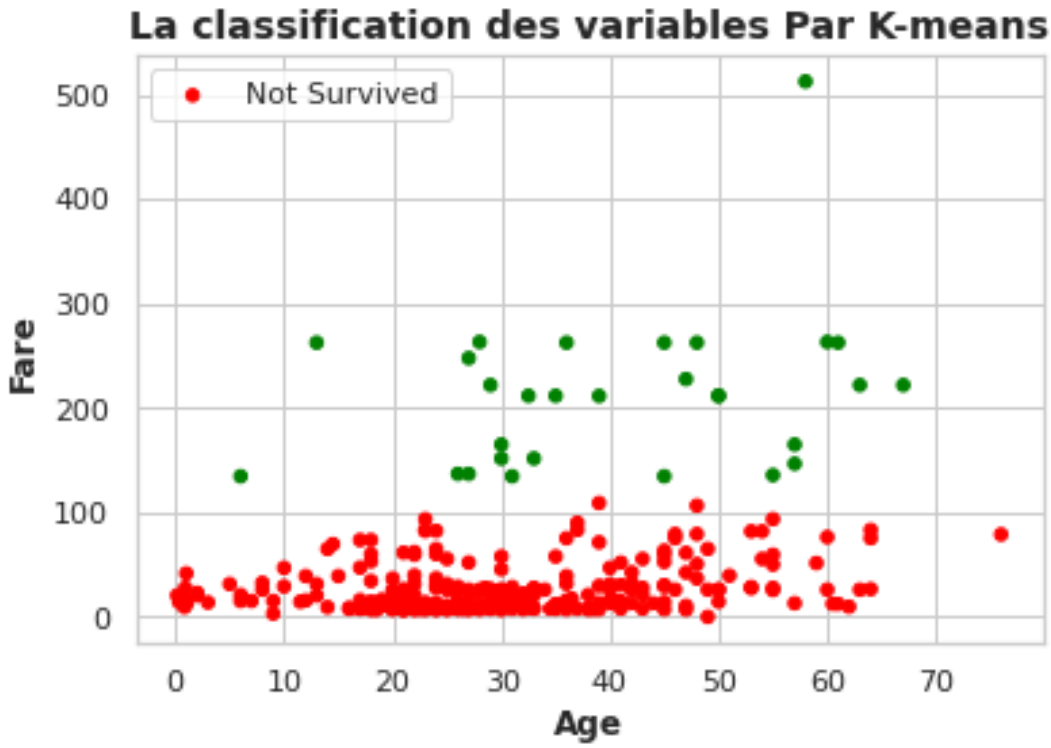
```
     Survived  Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
0           0       3    1  34.5      0      0    7.8292         2
1           1       3    0  47.0      1      0    7.0000         0
2           0       2    1  62.0      0      0    9.6875         2
3           0       3    1  27.0      0      0    8.6625         0
4           1       3    0  22.0      1      1   12.2875         0
..        ...     ...  ...   ...    ...    ...       ...       ...
327         1       3    0   3.0      1      1   13.7750         0
328         1       1    0  37.0      1      0   90.0000         2
329         1       3    0  28.0      0      0    7.7750         0
330         1       1    0  39.0      0      0  108.9000         1
331         0       3    1  38.5      0      0    7.2500         0

[332 rows x 8 columns]
```

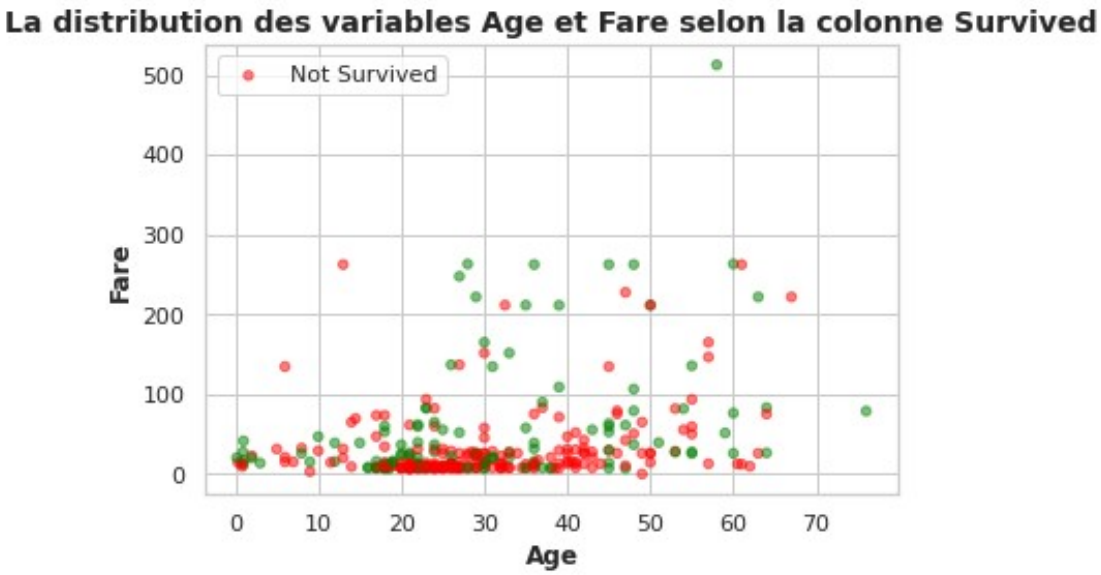
```
# import package
from sklearn.cluster import KMeans
```

```
#Application de KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(data_class)

#Visualisation
colormap=np.array(["red", "green"])
plt.scatter(data_class.Age, data_class.Fare,
c=colormap[kmeans.labels_], s=20)
plt.legend(['Not Survived'])
plt.title('La classification des variables Par K-means', fontsize=14,
fontweight='bold')
plt.xlabel('Age', fontweight='bold')
plt.ylabel('Fare',  fontweight='bold')
plt.show()
```




```
# affichage du nuage de point
plt.scatter(data_class.Age, data_class.Fare,
c=colormap[data_class.Survived], s=20, alpha=.5)
plt.legend(['Not Survived'])
plt.title('La distribution des variables Age et Fare selon la colonne
Survived', fontsize=14, fontweight='bold')
plt.xlabel('Age', fontweight='bold')
plt.ylabel('Fare',  fontweight='bold')
plt.show()
```



=> On obtient presque le même graphe sauf pour k-means, il considère la distance entre les points et le centre de gravité ce qui explique la petite différence.

Conclusion

En conclusion, on peut affirmer l'hypothèse logique que la majorité des personnes survivant sont des femmes de première classe et ceci est dû peut être au fait que les gens de première classe reçoivent l'aide en premier .

On peut conclure aussi que le facteur `Age` ne peut pas affirmer si la personne va survivre ou non

---

Références

- Le data set
- ACP
- AFC
- Cours analyse de données

This project is done by:

Ismail Ouahbi

Hamza Khalid