This project is done by:

Ismail Ouahbi

Hamza Khalid

Introduction

---

1. problème

On April 15, 1912, during its maiden voyage, the Titanic sank after striking an iceberg, killing 1,502 of 2224 passengers and crew. Survival rate was 32%. One of the reasons the sinking caused such a loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some factor of luck in surviving the sinking, some groups of people were more likely to survive than others, such as women, children and the upper class. As a result, the Titanic trip generated a lot of data that statisticians collected for analysis and enhancement to predict other situations and avoid the recurrence of such a problem .

---

2. Le data set

The data set is composed of 12 columns and 418 rows where 7 columns are quantitative and 5 qualitative, it has as null values 414 distributed between several columns

A brief description of each column is as follows:

· PassengerId - Single Passenger ID Survived - Survival Flag (0 = Dead, 1 = Survival) Pclass - Ticket Class Name - Passenger Name Sex - Gender (Male = Male, Female = Female) Age - Age SibSp - Number of siblings/spouses on board the Titanic Parch - Number of parents/children on the Titanic Ticket - Fare ticket number – Cabin fare – Embarked room number - Departure point (port on Titanic)

We will also give a brief description of each variable. pclass = ticket class 1 = upper class (rich) 2 = middle class (general class) 3 = lower class (working class)

Embarked = The definition of each variable is C = Cherbourg Q = Queenstown S = Southampton NaN = represents a data loss.

---

3. Assumptions and Questions

I can barely remember when I first watched the movie Titanic, but Titanic is still a topic of discussion in a wide variety of fields. Thus several questions arise at this point:

- What kind of people survive?
- What are the factors influencing people's survival?
- The age of people survives?
- What class dominates the survivor classes?
- Can we say that women and children have a strong chance of surviving?

##I. Data Mining

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

#load the dataset
data = pd.read_csv('tested.csv')

#data types
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

#predict data format
data.shape

(418, 12)

#read the first 5 lines
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
```

```
# data statistics
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. Data Preprocessing

```
#see the number of null values for each column
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
```

=> We can see that the three columns `Age`, `Fare` and `Cabin` have zero values

```
#see total number of null values
data.isnull(). sum(). sum()
```

```
414
```

=> The total number of null values is `414`

- **We will process the missing data:**

```
#For column 'Fares'
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
```

=> As this passenger is class 3 we will replace the missing Fare `column value by` the average `of the Fare` values of `the 3rd class`

```
#average `Fare' values of 3rd class avg_fare_p3 =
np.mean(data[data[ 'Pclass'] == 3]['Fare']) data['Fare'].
fillna(avg_fare_p3 , inplace=True)
```

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

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

```
#For 'Age' column
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]
```

=> As the Cabin `column` contains `327 null value` we will take a copy of the original dataset (to preserve it) and work with the original dataset where this column will be deleted

```
#take a copy
copie_data = data.copy()
#remove unwanted column
data.drop(['Cabin'] , axis=1 , inplace = True)

#verify
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
```

- **Remove missing values from the Age column:**

```
#all missing values in the age column represent only
20%
np.round((data.isnull().sum()['Age']/data.shape[0])*100,2)

20.57
```

```
# as the set of missing values in the age column represents
only 20% of the set of values
# we're going to cut that slice
data.drop(list(data[data['Age'].isnull()]['Age'].index) , axis = 0 ,
inplace=True)

# reenitialize the indices
data.reset_index(drop=True, inplace=True)

# verification
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
```

```
# know the number of rows repeated
data.duplicated(). sum()
```

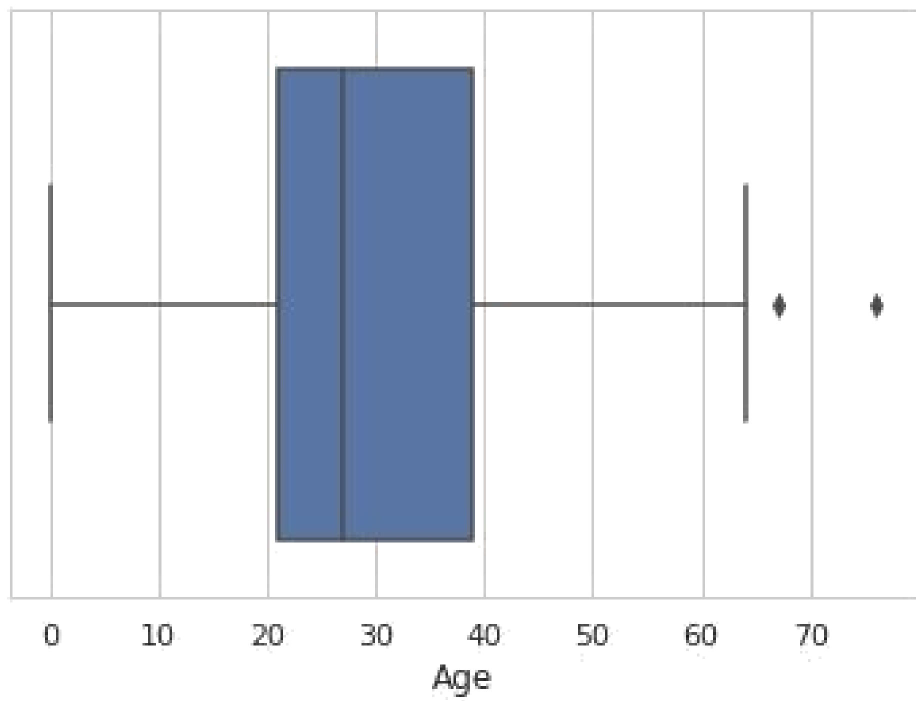
```
0
```

=> The dataset does not contain repeated values

- **Detection of outliers:**

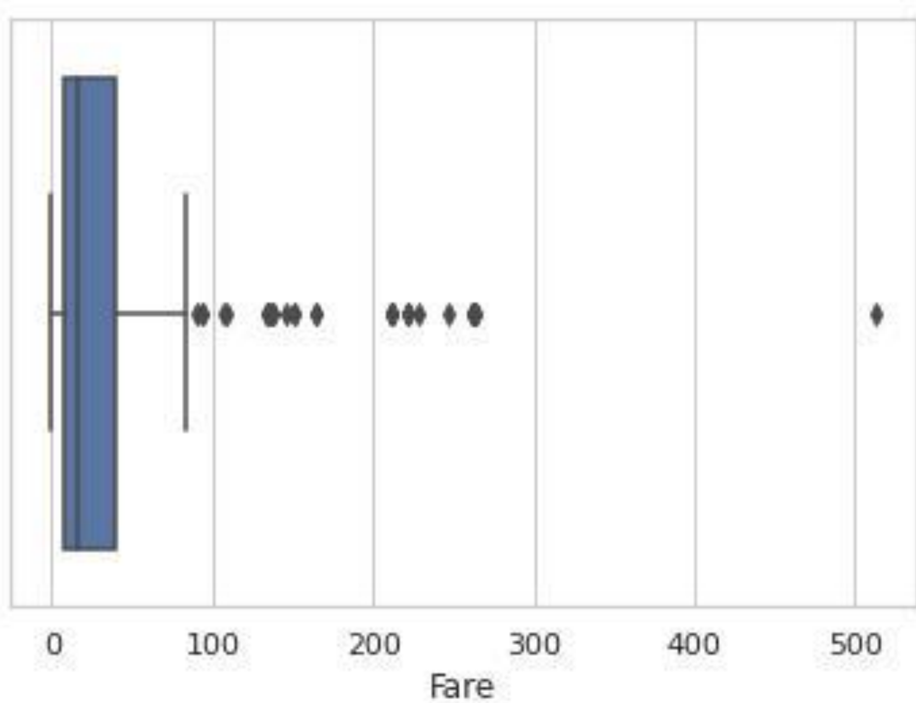
1. **For the Age variable :**

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



1. **For the Fare variable**:

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



=> The removal of `outliers` is not always the ideal solution, in our case the variables `Age` and `Fare` carry a lot of information and so we will keep these `outliers` .

III. Analysis Part:

---

1. ACP :

- **To be able to apply the main component analysis, we just need to take the quantitative variables:**

```
# choose numeric columns
numerical_col = data.dtypes[data.dtypes != object].index.tolist()

# a dataframe that contains only numeric columns
numerical_data = data[numerical_col]

#verifying
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

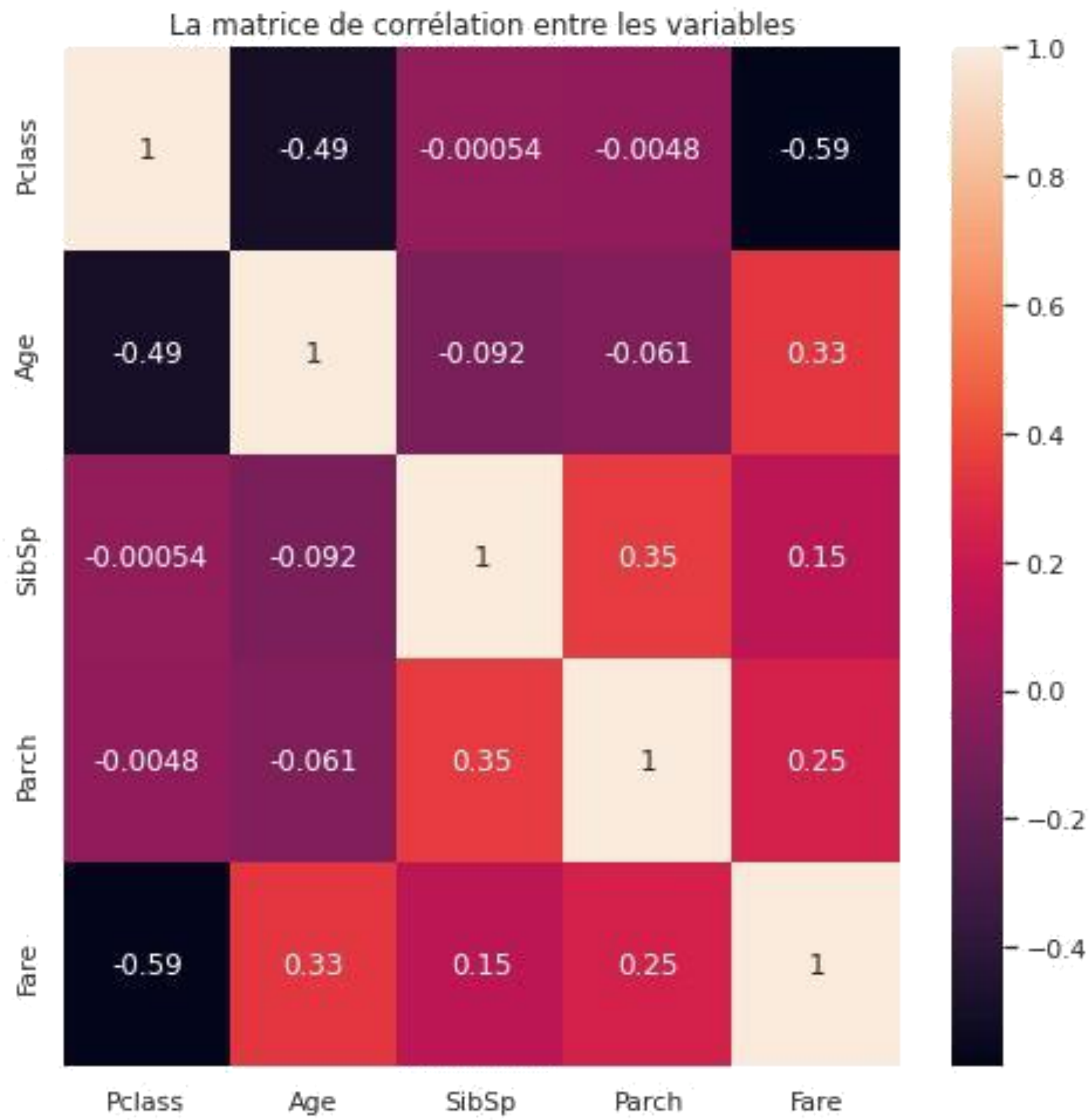
```
# removal of unnecessary numeric columns
numerical_data.drop(['PassengerId','Survived'], axis=1,inplace=True)
```

- **Correlation Matrix (Heatmap):**

```
#import required packages
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('The correlation matrix between variables')
plt.show()
```



=> The correlation between the varibales `Age`<=>`Pclass` is negative **=>** The correlation between the varibales `Pclass`<=>`Fare` is negative

=> For the rest of the correlations between the variables, the correlation varies between low and medium correlation.

- **Center and Reduce Data:**

```
#import required packages
from sklearn.preprocessing import StandardScaler

# separate features and target
#features
X = numerical_data
# Separating out the target
y = data[['Survived']]
# Standardizing the features
```

```
X_sc = StandardScaler().fit_transform(X)
# convert data to DataFrame
X_sc = pd.DataFrame(X_sc , columns = X.columns)

#verifying
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
31.012325 -0.231118 -0.552184 -0.491199 -0.527868
41.012325 -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]
```

- **Applying the CPA:**

```
#import packages
from sklearn.decomposition import PCA
#fix the number of components in 2
pca = PCA(n_components=2) #adjustment

principalComponents = pca.fit_transform(X_sc)

#verify data
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['main component 1', 'main component
2'])
principalDf

             main component 1        main component 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]
```

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

#verification
finalDf
```

```
                main component 1          main component 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]
```

```
# The percentage of information guaranteed by each component
pca.explained_variance_ratio_
```

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

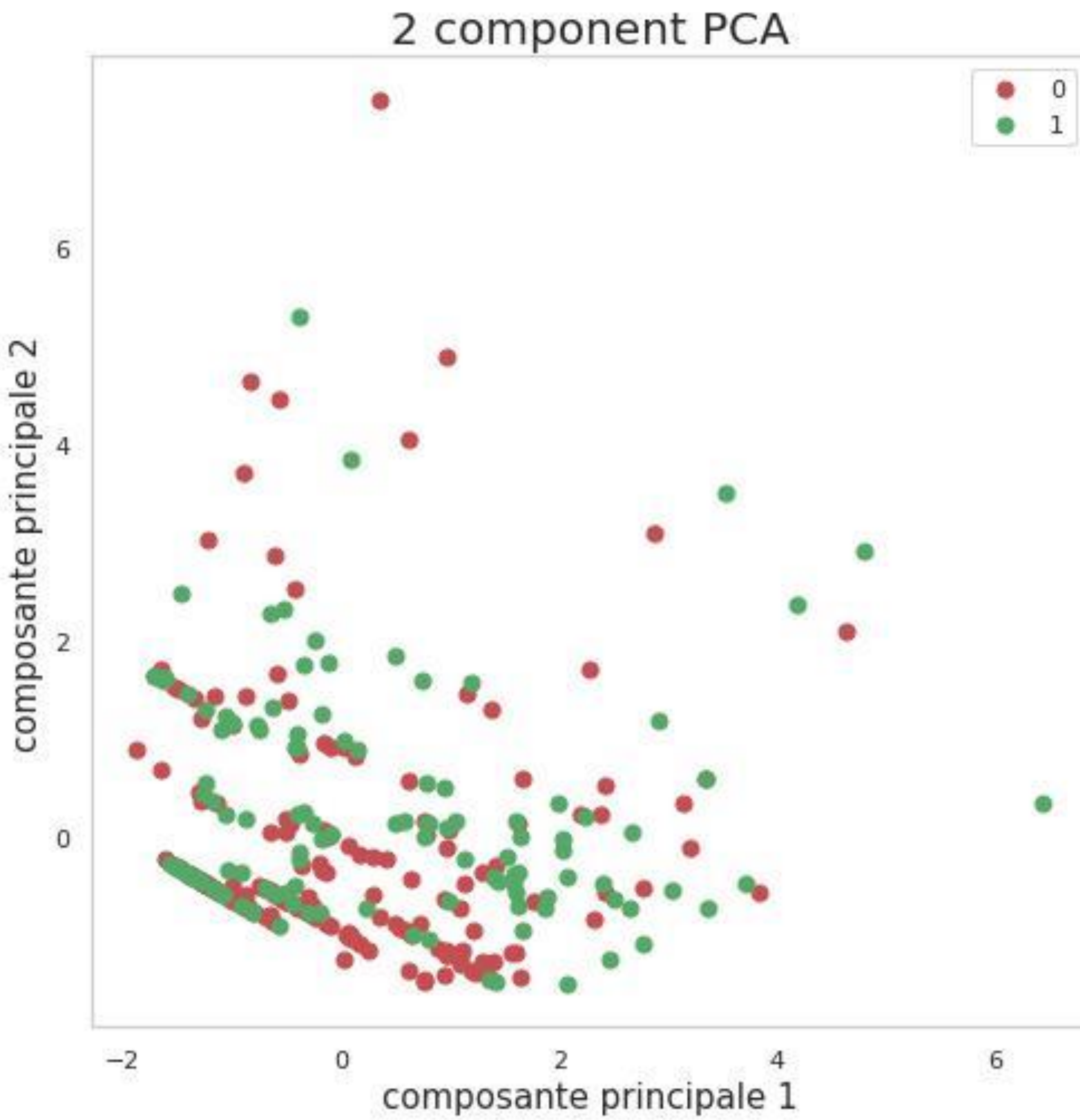
=> We note that the first component contains the 39.41% information while the second 28.59% of information which gives 68% information in the new space

- ***Visualization of components (ACP):***

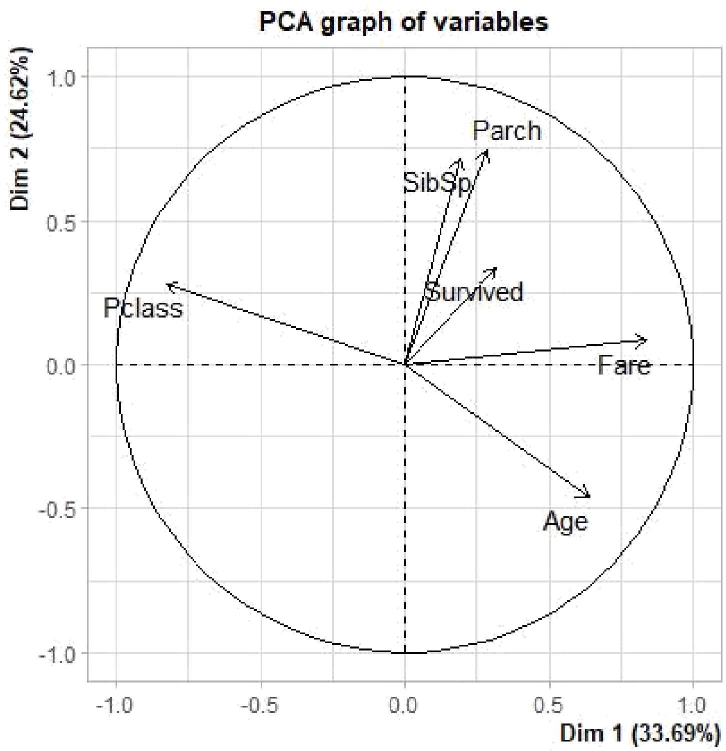
1. **With Python:**

```
# adjust window dimensions fig =
plt.figure(figsize = (8,8))
# add the sub-plot
ax = fig.subplots(1,1)
# Label the axes
ax.set_xlabel('component  principale 1', fontsize = 15)
ax.set_ylabel('component  principale 2', fontsize = 15)
# added title
ax.set_title('2 component PCA', fontsize = 20)
# the variable containing the target target
values = [0,1]
# pitch colours
colors = ['r', 'g']
# process
for target, color in zip(targets,colors): indicesToKeep =
    finalDf['Survived'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'composante principale 1']
               , finalDf.loc[indicesToKeep, 'main component 2']
               , c = color
```

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



1. **With R:**



=> The correlation between varibales `Age`<=>`Pclass` is ***negative* =>**

SibSp<=>Parch<=>Survived `is strong` => ***Age***<=>`Parch` varibales are independent

=> Survived<=>Age `varibales are independent`

=> Survived<=>Pclass `varibales are independent`

2. AFC :

```
# we will take a copy of the dataset to perform the AFC
afc_data = data[['Sex', 'Embarked', 'Pclass', 'Age','Survived']]

# afc_data.head
() verification
```

	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

- **As the variable `Age` has several values, we will group them according to bins `intervals`, to better see the data:**

```
# see statistics on the 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
```

```
# deviation of 'Age' values
afc_data['Age_bin'] = pd.cut(afc_data['Age'],
                             [0, 21, 27, 39, 76])

# display
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
```

- **Since the variable `Pclass` has several values(1,2,3...), we will make them qualitative (first,second,third...):**

```
# see 'Pclass' variable modalities
afc_data['Pclass'].value_counts().sort_index()
```

```
1     98
2     88
3146
Name: Pclass, dtype: int64
```

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

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

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

- **As the variable `Survived` has two values(0,1), we will make them qualitative(survived,not survived):**

```
# see the modalities of the variable 'Survives'
afc_data['Survived'].value_counts().sort_index()
```

```
0    205
1127
Name: Survived, dtype: int64
```

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

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

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

- **Apply the AFC:**

```
# display of the first 2 lines
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  the necessary package
import prince
```

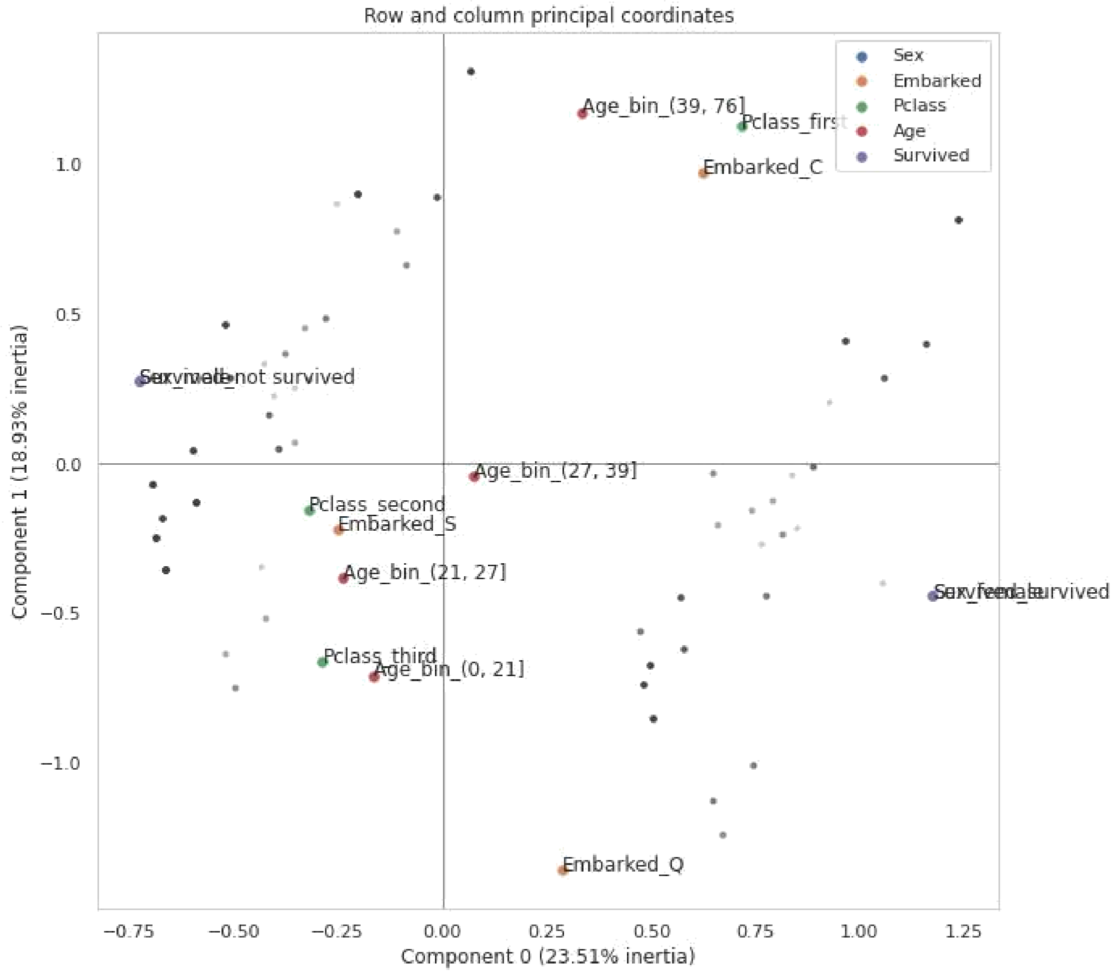
```
#take the dataset
afc_data = afc_data[['Sex', 'Embarked', 'Pclass',
'Age_bin','Survived']]
```

```
#instantiate the model
mca = prince.MCA()
mca.fit(afc_data)
```

```
MCA()
```

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

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



=> The gender of the surviving people is `female` because the distance between the points representing these entities is too small, while the non-survivant people are usually `male` .

=> People between the ages of 39 and 76 dominate the first class whose starting point is `C` .

=> People whose starting point is S have chosen the `2nd class` .

=> We also note that the Age `variable` has no correlation with the Survived variable .

3. K-mean Classification

```
# take a copy of the
data_class dataset =
data.copy()

# verification
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
```

```
# make the "Sex" variable relevant
data_class['Sex'].replace(['male','female'], [1,0] , inplace=True)

# make the "Embarked" variable relevant
data_class['Embarked']. replace(['S','C','Q'], [0,1,2], inplace=True)

# filtering
and           display
data_class = data_class[['Survived','Pclass','Sex',
'Age','SibSp','Parch','Fare','Embarked']
data_class
```

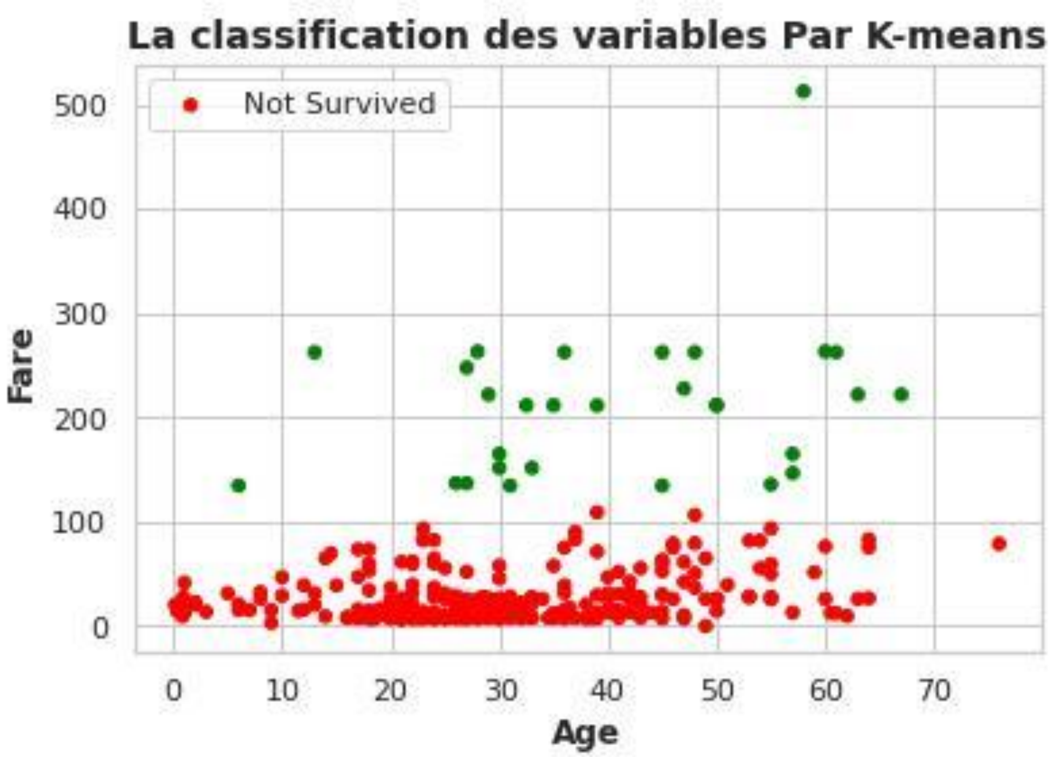
```
     Survived  Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
0           0       3    1  34.5      0      0    7.8292         2
1           1       3    0  47.0      1      0    7.0000         0
2           0       2    1  62.0      0      0    9.6875         2
3           0       3    1  27.0      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]
```

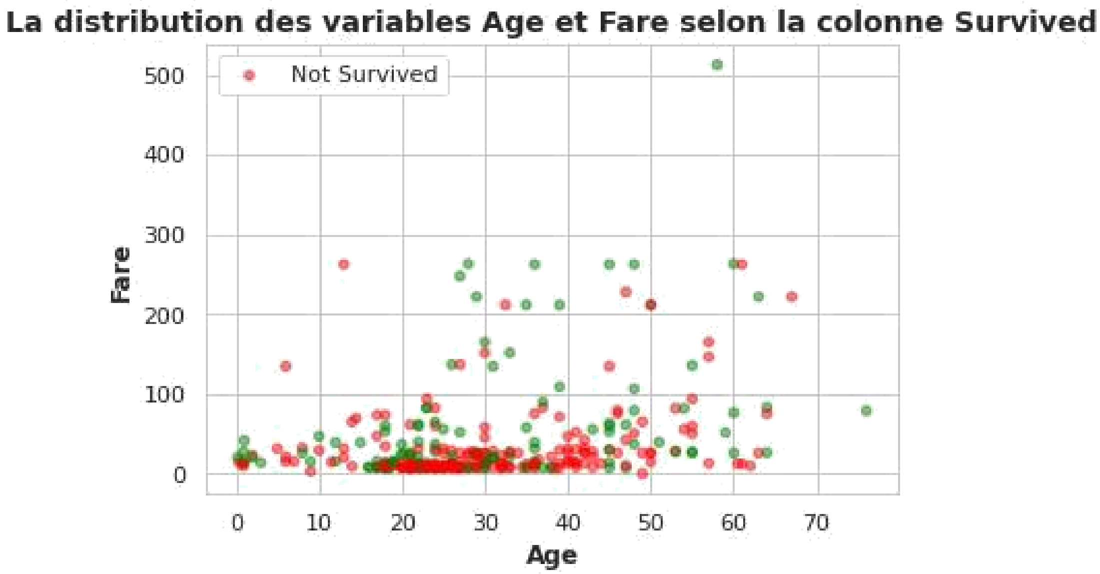
```
# import package
from sklearn.cluster import KMeans
```

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

#Viewing
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()
```




```
# display of point cloud
plt.scatter(data_class.Age, data_class.Fare,
c=colormap[data_class.Survived], s=20, alpha=.5)
plt.legend(['Not Survived'])
plt.title('The distribution of the Age and Fare variables according
to the Survived column', fontsize=14, fontweight='bold')
plt.xlabel('Age', fontweight='bold')
plt.ylabel('Fare', fontweight='bold')
plt.show()
```



=> We get almost the same graph except for k-means, it considers the distance between the points and the center of gravity which explains the small difference.

Conclusion

In conclusion, the logical assumption can be made that the majority of survivors are first-class women and this may be due to the fact that first-class people receive help first .

It can also be concluded that the Age `factor` cannot say whether the person will survive or not

---

References

- Le data set
- ACP
- AFC
- Data Analysis Course

This project is done by:

Ismail Ouahbi

Hamza Khalid