Projeqtor

Analyse descriptive

Entrée [5]:

```
# Importation des librairies

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.decomposition import PCA
import sqlite3
from sqlite3 import OperationalError
import psycopg2
from pandas import DataFrame
```

Entrée [179]:

```
conn = psycopg2.connect(host='localhost',port='5432',database='projeqtordwsql3', user='post
```

Entrée [180]:

```
cursor = conn.cursor()
cursor.execute("""SELECT * FROM fact_project""")
query_results = cursor.fetchall()
df = DataFrame (query_results,columns=['id','idProject','Phase','Resource','Client','Assigr
print (df)
```

```
       id  idProject  Phase  Resource  Client  Assignment  CreationDate  
\
0      59          6     35        12      11        2056      20180219
1      60          6     36        12      11        2056      20180219
2      61          6     37        12      11        2056      20180219
3      62          6     38        12      11        2056      20180219
4    1709         43    238        15      24         291      20180227
5    1710         43    238        15      24         419      20180227
6    1711         43    238        15      24        1022      20180227
7    1712         43     78        27      24         879      20180227
8    1713         43     78        27      24         883      20180227
9    1714         43     78        27      24         885      20180227
10   1715         43     78        27      24         888      20180227
11   1716         43     78        27      24         891      20180227
12   1717         43     79        27      24         879      20180227
13   1718         43     79        27      24         883      20180227
14   1719         43     79        27      24         885      20180227
15   1720         43     79        27      24         888      20180227
16   1721         43     79        27      24         891      20180227
```

Entrée [181]:

```
cursor2 = conn.cursor()
cursor2.execute("""SELECT * FROM dim_bill""")
query_results2 = cursor2.fetchall()
df2 = DataFrame (query_results2,columns=['ID_BILL','id','billingType','paymentDone','paymer
print (df2)
```

```
73             0
74             0
75             0
76             0
77             0
78             0
79             0
80             0
81             0
82             0
83             0
84             0
85             0
86             0
87             0
88             0
89             0
90             0
91             0
92             0
```

Entrée [178]:

```
df.dtypes
```

Out[178]:

```
idProject            int64
Phase                int64
Resource             int64
Client               int64
Assignment           int64
CreationDate         int64
Bill                 int64
Sector               int64
Project_duration     int64
Delay_Assignment     int64
Delay_Total          int64
NbProject            int64
NbPhases             int64
dtype: object
```

Entrée [145]:

```
#verification des valeurs manquantes
df.isnull().sum()
```

Out[145]:

```
id                   0
idProject            0
Phase                0
Resource             0
Client               0
Assignment           0
CreationDate         0
Bill                 0
Sector               0
Project_duration     0
Delay_Assignment     0
Delay_Total          0
NbProject            0
NbPhases             0
dtype: int64
```

Entrée [146]:

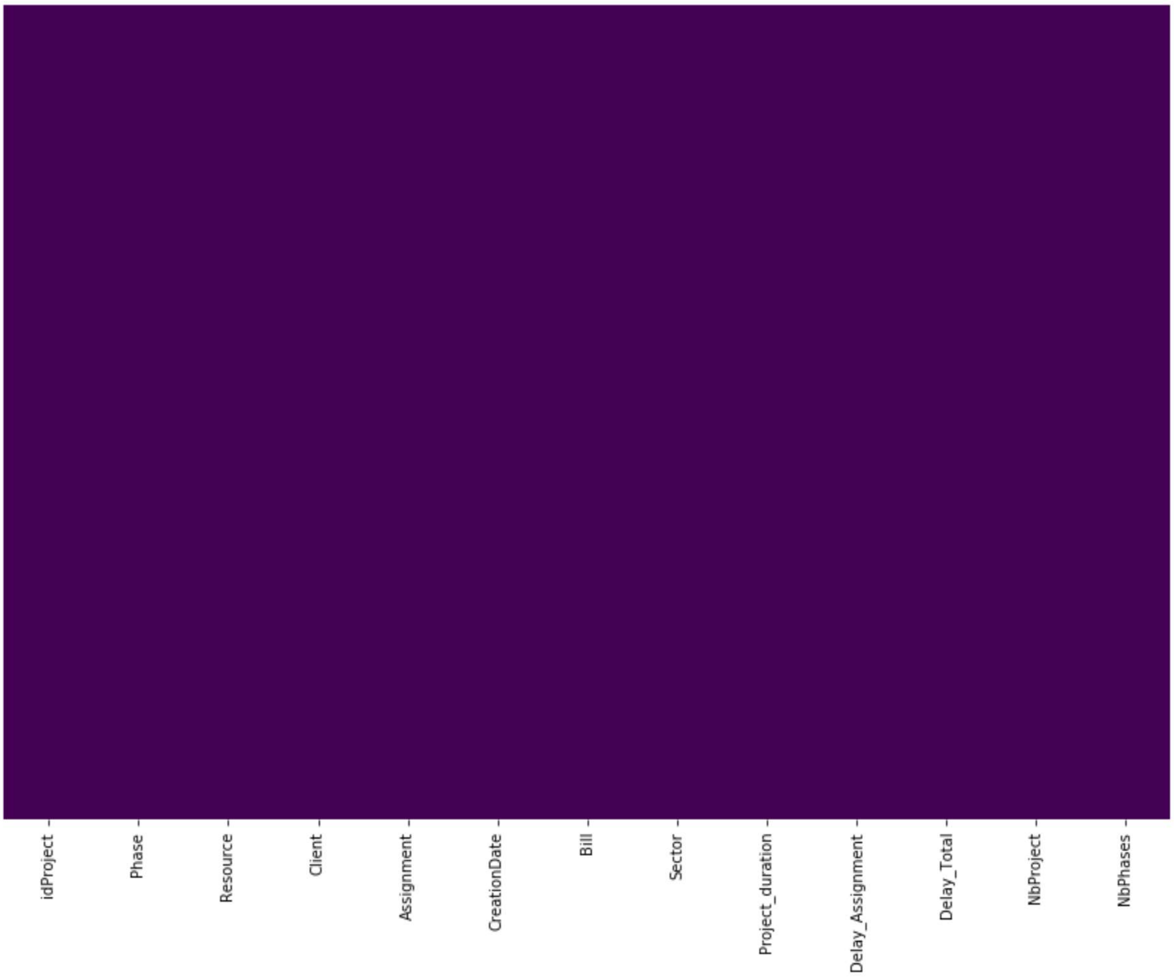
```
# creation une liste qui contient les colonne que je veut les supprimer
deleted_columns=['id']
df.drop(deleted_columns,axis=1,inplace=True)
```

Entrée [147]:

```
#verification des valeurs manquantes
plt.figure(figsize=(14,10))
sns.heatmap(df.isnull(),yticklabels=False, cbar=False,cmap='viridis')
```

Out[147]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23c66c28d68>
```



Entrée [148]:

```
print(df.shape)
```

```
(9375, 13)
```

Entrée [149]:

```
df.describe()
```

Out[149]:

	idProject	Phase	Resource	Client	Assignment	CreationDate	
count	9375.000000	9375.000000	9375.000000	9375.000000	9375.000000	9.375000e+03	9375.00
mean	69.534827	275.936960	26.577813	29.846720	1220.066880	2.018907e+07	69.53
std	19.902478	125.845589	17.244168	15.091834	539.228787	8.657508e+03	19.90
min	1.000000	1.000000	4.000000	1.000000	23.000000	2.018022e+07	1.00
25%	56.000000	167.000000	12.000000	15.000000	818.000000	2.018051e+07	56.00
50%	67.000000	264.000000	25.000000	32.000000	1205.000000	2.019032e+07	67.00
75%	88.000000	389.000000	40.000000	44.000000	1730.000000	2.020060e+07	88.00
max	102.000000	485.000000	63.000000	56.000000	2080.000000	2.021012e+07	102.00

Entrée [150]:

```
# Nettoyage des donnees
# si existe des lignes repetes les supprimer
df.duplicated().sum()
```

Out[150]:

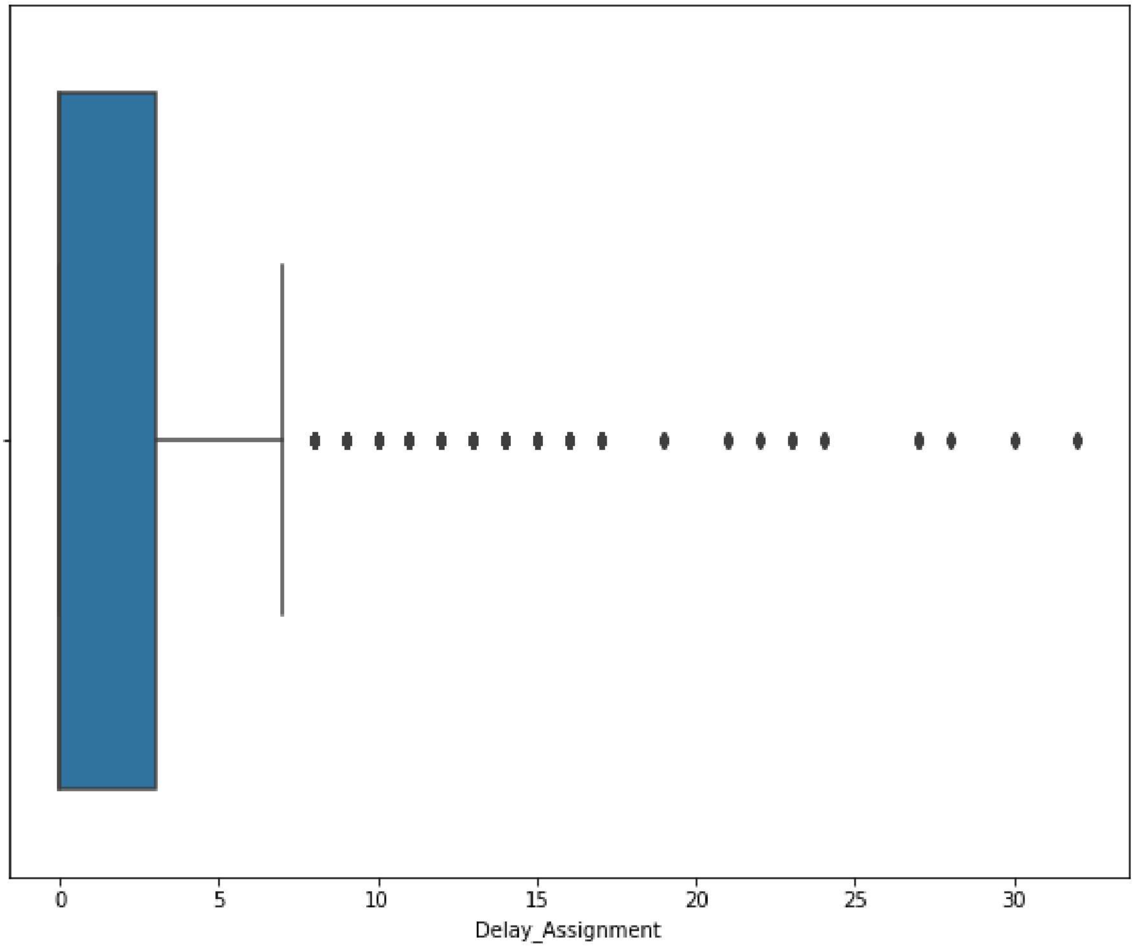
```
36
```

Entrée [151]:

```
#boxplot
plt.figure(figsize=(10,8))
sns.boxplot(x=df['Delay_Assignment'])
```

Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23c4311d358>
```



Entrée [ ]:

```

```

Entrée [152]:

```
#detecter les valeurs aberrantes en utilisant z_score
outliers=[]
def detect_outliers(data):
    threshold=3
    mean=np.mean(data)
    std=np.std(data)
    for i in data:
        z_score=(i-mean)/std
        if np.abs(z_score)>threshold:
            outliers.append(i)
    return outliers
```

Entrée [ ]:

```

```

Entrée [153]:

```
#detecter les delayAssignment consideres comme outliers
detect_outliers(df['Delay_Assignment'])
```

```
 30,
 30,
 30,
 16,
 16,
 16,
 16,
 16,
 16,
 16,
 16,
 27,
 16,
 16,
 16,
 16,
 16,
 16,
 16,
 16,
```

Entrée [ ]:

```

```

Entrée [154]:

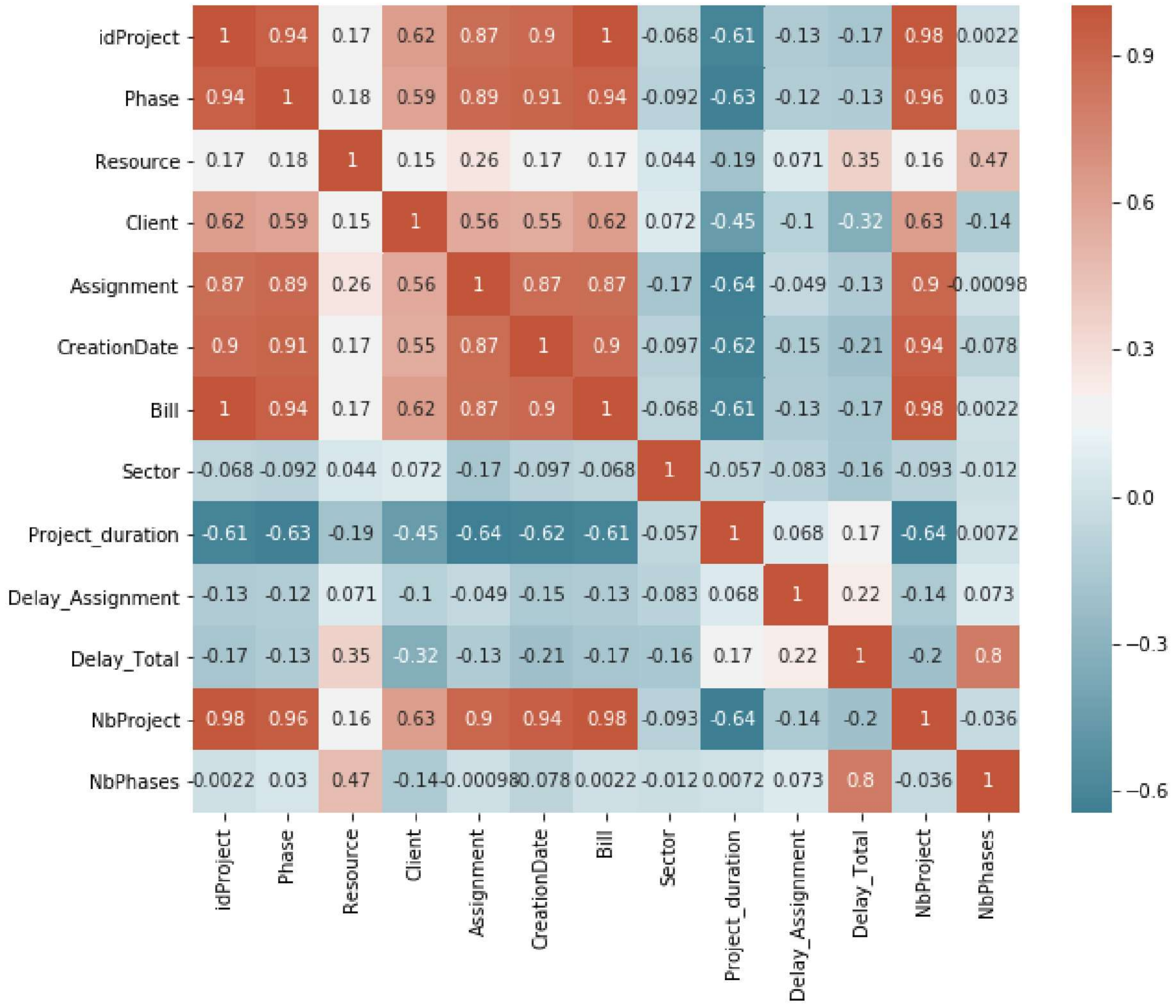
```
indexNames2 = df[ df['Delay_Assignment'] > 16 ].index

# Delete these row indexes from dataFrame
df.drop(indexNames2 , inplace=True)
```

Entrée [155]:

```
corr=df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns, annot=True,cmap=sns.div
```

Out[155]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23c430b5ac8>
```



Entrée [156]:

```
# relation bivariée entre le client et le nombre de projets
newDF=df[['Client','NbProject']]
newDF.head()
```

Out[156]:

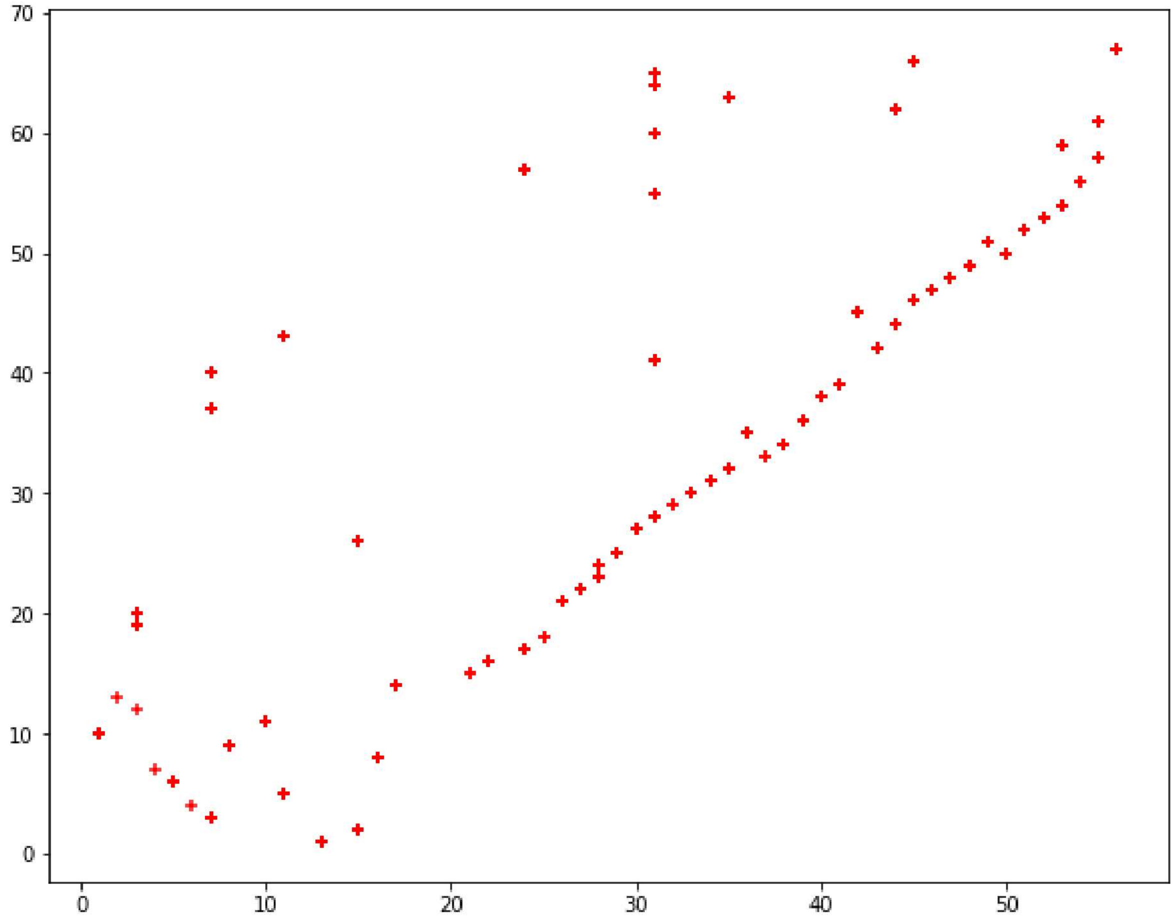
	Client	NbProject
0	11	5
1	11	5
2	11	5
3	11	5
4	24	17

Entrée [157]:

```
plt.figure(figsize=(10,8))
plt.scatter(df.Client,df.NbProject,marker='+',color='red')
```

Out[157]:

```
<matplotlib.collections.PathCollection at 0x23c68912b70>
```



Entrée [168]:

```
# relation bivariée entre l'Assignment et le Delay_Assignment
newDF=df[['Assignment','Delay_Assignment']]
newDF.head()
```

Out[168]:

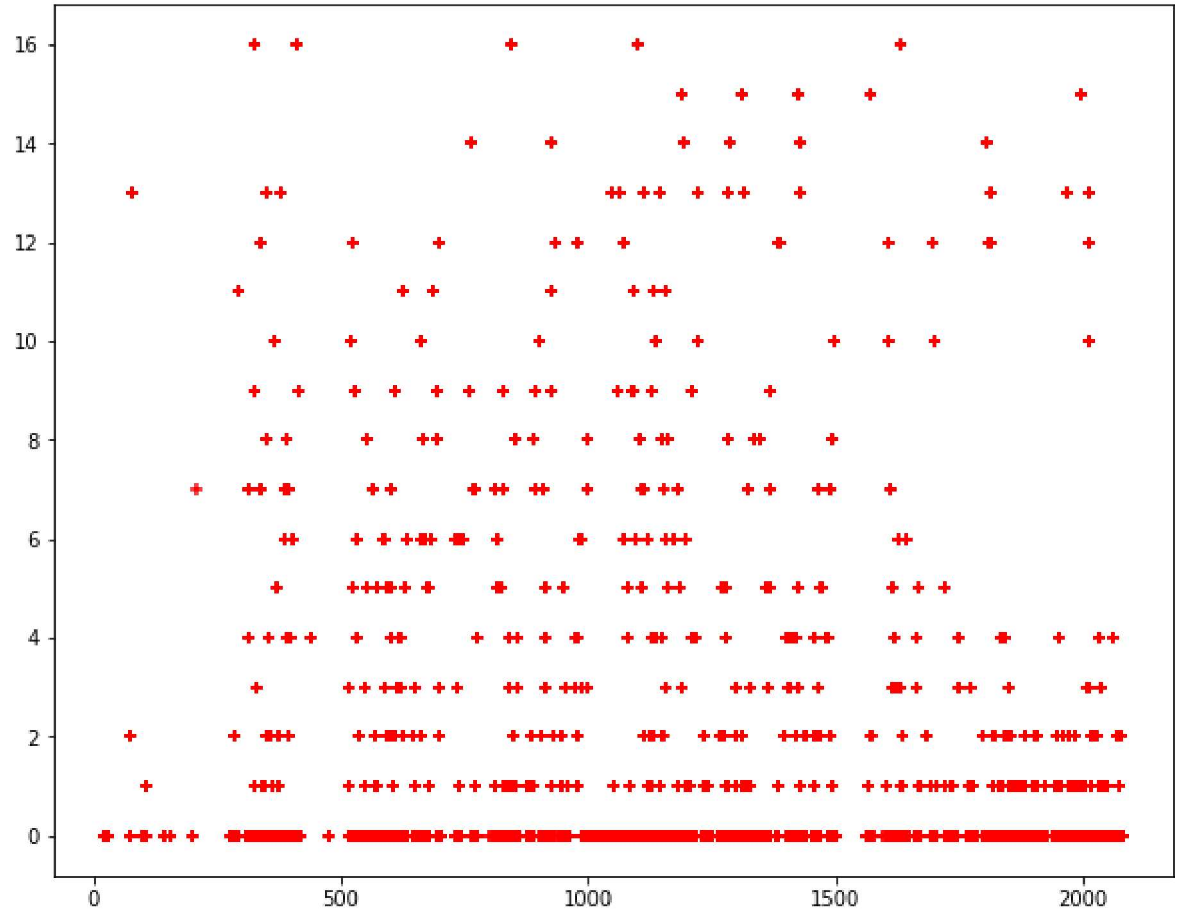
	Assignment	Delay_Assignment
0	2056	0
1	2056	0
2	2056	0
3	2056	0
4	291	0

Entrée [170]:

```
plt.figure(figsize=(10,8))
plt.scatter(df.Assignment,df.Delay_Assignment,marker='+',color='red')
```

Out[170]:

```
<matplotlib.collections.PathCollection at 0x23c6ab72860>
```

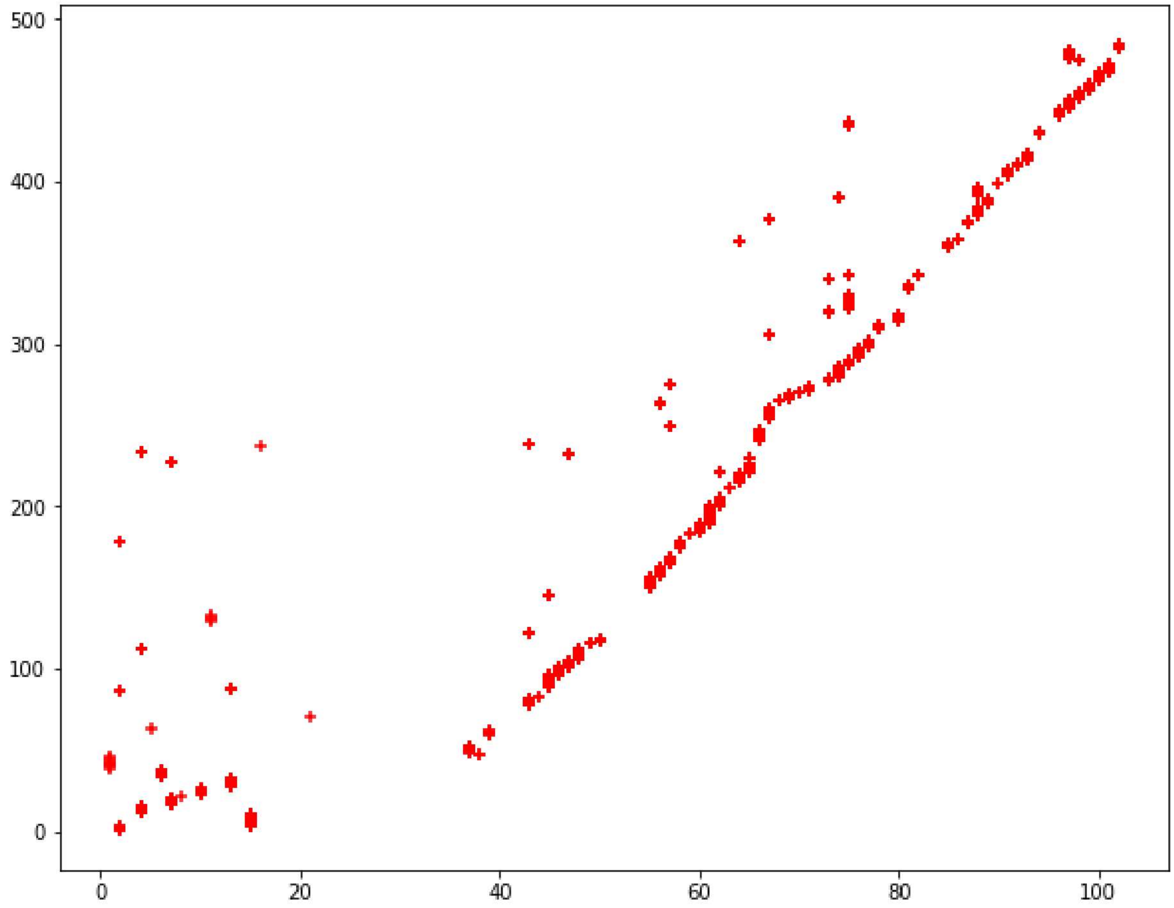


Entrée [167]:

```
# relation bivariée entre le projet et la phase
plt.figure(figsize=(10,8))
plt.scatter(df.idProject,df.Phase,marker='+',color='red')
```

Out[167]:

```
<matplotlib.collections.PathCollection at 0x23c6a311978>
```



Entrée [161]:

```
```

Out[161]:

```
<Figure size 1008x720 with 0 Axes>

<Figure size 1008x720 with 0 Axes>
```

Entrée [ ]:

```
```

Entrée [218]:

```
# relation bivariée entre le projet et le Bill
newDF=df2[['idProject','paymentDone']]
newDF.head()
```

Out[218]:

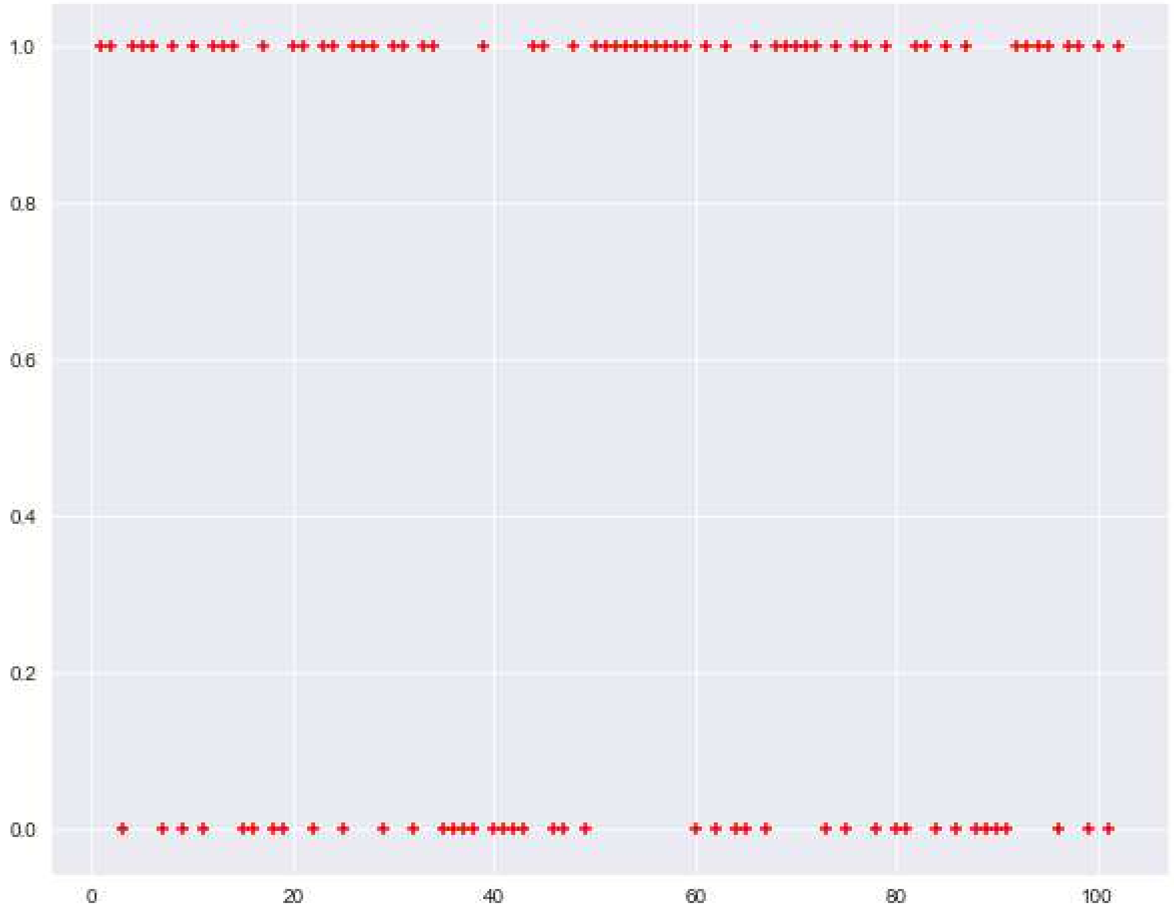
	idProject	paymentDone
0	29	0
1	30	1
2	31	1
3	32	0
4	33	1

Entrée [219]:

```
plt.figure(figsize=(10,8))
plt.scatter(df2.idProject,df2.paymentDone,marker='+',color='red')
```

Out[219]:

```
<matplotlib.collections.PathCollection at 0x23c772c42e8>
```

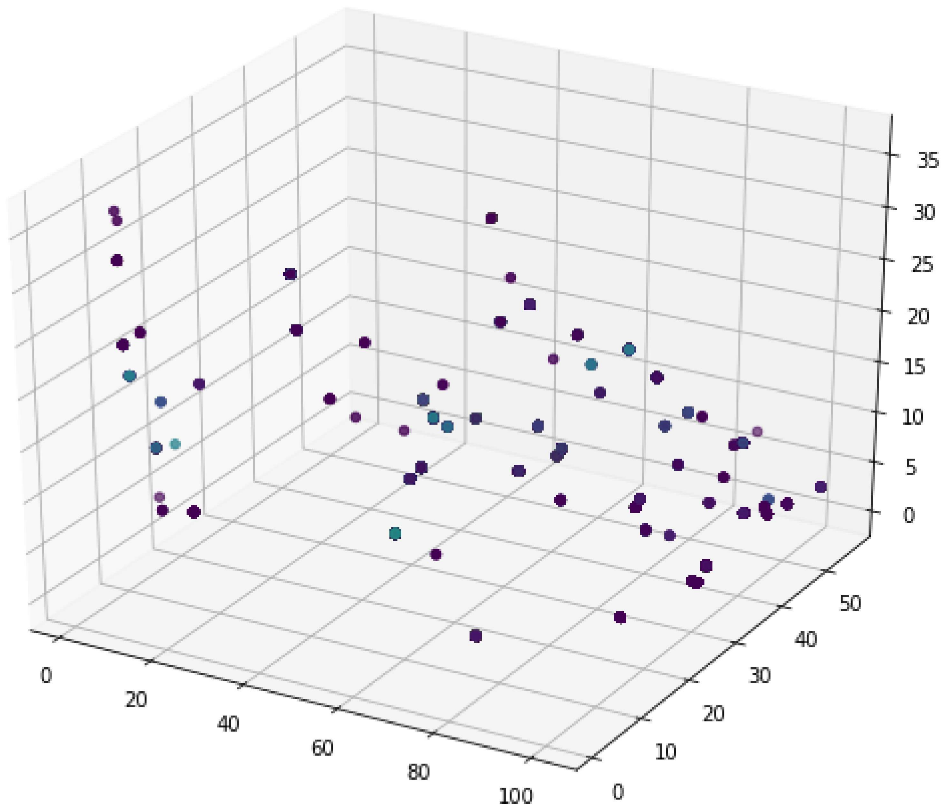


Entrée [174]:

```
# relation 3D entre projet, client, la durée et la dépassement
plt.figure(figsize=(10,8))
ax = plt.axes(projection='3d')
ax.scatter(df.idProject,df.Client,df.Project_duration, c=df.Delay_Assignment)
```

Out[174]:

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x23c6b0e6160>
```



Prédiction des projets payées

Entrée [186]:

```
model = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(df2[['idProject']],df2.paymentDone,trai
```

Entrée [187]:

```
X_test
```

Out[187]:

	idProject
3	32
52	73
98	99
89	90
62	83
20	69
64	85
88	89
29	9
90	91
11	40
48	28
61	82
13	42
54	75
51	72
86	67
91	92
95	96
44	24
16	45

Entrée [188]:

```
model.fit(X_train, y_train)
X_test
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[188]:

```
    idProject
 3         32
52         73
98         99
89         90
62         83
20         69
64         85
88         89
29          9
90         91
11         40
48         28
61         82
13         42
54         75
51         72
86         67
91         92
95         96
44         24
16         45
```

Entrée [189]:

```
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
```

Out[189]:

```
array([[0.38874144, 0.61125856],
       [0.35094991, 0.64905009],
       [0.32788567, 0.67211433],
       [0.33578267, 0.66421733],
       [0.3419892 , 0.6580108 ],
       [0.35456426, 0.64543574],
       [0.34021027, 0.65978973],
       [0.33666592, 0.66333408],
       [0.41057684, 0.58942316],
       [0.33490058, 0.66509942],
       [0.38124522, 0.61875478],
       [0.39250962, 0.60749038],
       [0.34288034, 0.65711966],
       [0.37937982, 0.62062018],
       [0.3491491 , 0.6508509 ],
       [0.35185191, 0.64814809],
       [0.35637771, 0.64362229],
       [0.33401964, 0.66598036],
       [0.33050748, 0.66949252],
       [0.39629065, 0.60370935],
       [0.3765884 , 0.6234116 ]])
```

Entrée [190]:

```
model.score(X_test,y_test)
```

Out[190]:

```
0.42857142857142855
```

Entrée [191]:

```
y_predicted
```

Out[191]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      dtype=int64)
```

Entrée [192]:

```
X_test
```

Out[192]:

	idProject
3	32
52	73
98	99
89	90
62	83
20	69
64	85
88	89
29	9
90	91
11	40
48	28
61	82
13	42
54	75
51	72
86	67
91	92
95	96
44	24
16	45

Entrée [193]:

```
model.coef_
```

Out[193]:

```
array([[0.00395758]])
```

Entrée [194]:

```
model.intercept_
```

Out[194]:

```
array([0.32596307])
```

Entrée [195]:

```
import math
def sigmoid(x):
  return 1 / (1 + math.exp(-x))
```

Entrée [198]:

```
def prediction_function(idProject):
    z = 0.042 * idProject - 1.53 # 0.04150133 ~ 0.042 and -1.52726963 ~ -1.53
    y = sigmoid(z)
    return y
```

Entrée [199]:

```
idProject = 13
prediction_function(idProject)
```

Out[199]:

```
0.2720988176402692
```

Entrée [200]:

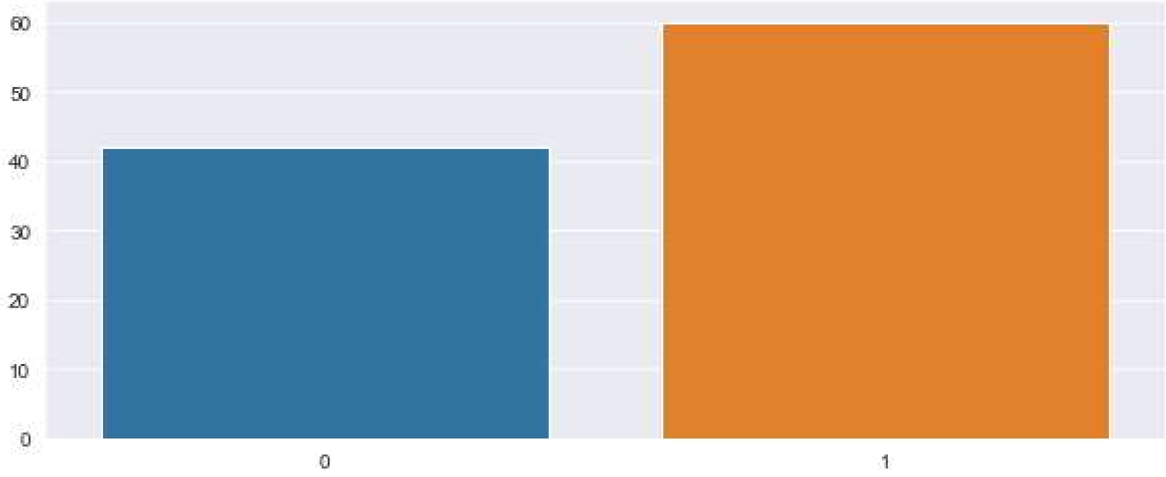
```
idProject = 34
prediction_function(idProject)
```

Out[200]:

```
0.474522085522507
```

Entrée [202]:

```
genders = df2.paymentDone.value_counts()
genders2 = df2.idProject.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()
```



classification:

Entrée [209]:

```
#Clustering
#KMeans
from sklearn.cluster import KMeans
clusters=5
kmeans = KMeans(n_clusters = clusters)
kmeans.fit(query_results)
print(kmeans.labels_)
print(kmeans.inertia_)
```

```
[1 1 1 ... 2 2 2]
17971713188.60685
```

Entrée [213]:

```
#EDA Analysis
#PCA
from sklearn.decomposition import PCA
pca = PCA(7)
pca.fit(query_results)

pca_data = pd.DataFrame(pca.transform(query_results))

print(pca_data.head())
```

```
             0            1           2            3           4           5  \
0 -9593.220068  2582.516644  -17.634231  1358.498096 -121.774483  223.859338
1 -9592.989752  2581.503445  -17.385587  1358.582334 -120.764050  222.965929
2 -9592.759435  2580.490247  -17.136943  1358.666571 -119.753616  222.072520
3 -9592.529118  2579.477049  -16.888299  1358.750809 -118.743183  221.179112
4 -9313.948507  1008.943178 -414.003851  -390.005568   92.346264 -117.956889

           6
0  41.654319
1  41.681813
2  41.709308
3  41.736802
4  -5.353247
```

Entrée [214]:

```
from matplotlib import colors as mcolors
import math

''' Generating different colors in ascending order
                                of their hsv values '''
colors = list(zip(*sorted((
                    tuple(mcolors.rgb_to_hsv(
                          mcolors.to_rgba(color)[:3])), name)
                     for name, color in dict(
                            mcolors.BASE_COLORS, **mcolors.CSS4_COLORS
                                                      ).items())))[1]


# number of steps to taken generate n(clusters) colors
skips = math.floor(len(colors[5 : -5])/clusters)
cluster_colors = colors[5 : -5 : skips]
```

Entrée [ ]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(pca_data[0], pca_data[1], pca_data[2],
           c = list(map(lambda label : cluster_colors[label],
                                            kmeans.labels_)))

str_labels = list(map(lambda label:'% s' % label, kmeans.labels_))

list(map(lambda data1, data2, data3, str_label:
        ax.text(data1, data2, data3, s = str_label, size = 16.5,
        zorder = 20, color = 'k'), pca_data[0], pca_data[1],
        pca_data[2], str_labels))

plt.show()
```

Entrée [ ]:

```
```