P10. Détectez des faux billets avec Python

Thi Tuong LE

Data Analytics Team

Nettoyage des données

- Sélection des données
- Traitement des valeurs manquantes

Analyse des données

- Réaliser une ACP
- Utiliser un algorithme de clustering de type Kmeans
- Modéliser grâce à la régression logistique
- Interpréter une ACP

Import des librairies

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from statsmodels.formula.api import ols
from statsmodels.stats.diagnostic import het_white , normal_ad
from sklearn.linear_model import LinearRegression

from statsmodels.compat import lzip
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels

from sklearn import decomposition
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

from matplotlib.collections import LineCollection
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn import decomposition, preprocessing
from sklearn import cluster, metrics
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale

from statsmodels.api import Logit
from statsmodels.genmod.generalized_linear_model import GLM
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score, precision_score, recall_score, f1_score, classification_report

from scipy.stats import t, shapiro
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Les fonctions

```
def display_scree_plot(pca):
    scree = pca.explained_variance_ratio_*100
    plt.bar(np.arange(len(scree))+1, scree)
    plt.plot(np.arange(len(scree))+1, scree.cumsum(),c="red",marker='o')
    plt.xlabel("rang de l'axe d'inertie")
    plt.ylabel("pourcentage d'inertie")
    plt.title("Eboulis des valeurs propres")
    plt.show(block=False)
```

Représenter graphiquement les Éboulis des valeurs propres

```
# Fonction qui me permet d'afficher les cercles de corrélations
def display_circles(pcs, n_comp, pca, axis_ranks, labels=None, label_rotation=0, lims=None
    for d1, d2 in axis_ranks: # On affiche les 3 premiers plans factoriels, donc les 6 pre
        if d2 < n_comp:

            # initialisation de la figure
            fig, ax = plt.subplots(figsize=(8,8))
```

Afficher les cercles de corrélations

```
#Fonction qui me permet de projeter mes individus sur un plan
def display_factorial_planes(X_projected, n_comp, pca, axis_ranks, labels=None, alpha=1, i
    for d1,d2 in axis_ranks:
        if d2 < n_comp:

            # initialisation de la figure
            fig = plt.figure(figsize=(7,6))
```

Projeter mes individus sur un plan

Import des données

- Billets_prodution.csv :

```
billets_prod = pd.read_csv("billets_production.csv")
```

```
billets_prod
```

	diagonal	height_left	height_right	margin_low	margin_up	length	id
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5

- billets.csv

```
bil = pd.read_csv("billets.csv", sep=";", decimal='.')
```

```
bil
```

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	True	171.81	104.86	104.95	4.52	2.89	112.83
1	True	171.46	103.36	103.66	3.77	2.99	113.09
2	True	172.69	104.48	103.50	4.40	2.94	113.16
3	True	171.36	103.91	103.94	3.62	3.01	113.51
4	True	171.73	104.28	103.46	4.04	3.48	112.54
...	...	...	...	...	...	...	...
1495	False	171.75	104.38	104.17	4.42	3.09	111.28

Information des paramètres

Length (mm)	Diagonal (mm)
height_left (mm)	height_right (mm)
margin_up (mm)	margin_low (mm)

Observation des données

```
billets_prod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   diagonal       5 non-null      float64
 1   height_left    5 non-null      float64
 2   height_right   5 non-null      float64
 3   margin_low     5 non-null      float64
 4   margin_up      5 non-null      float64
 5   length         5 non-null      float64
 6   id             5 non-null      object 
dtypes: float64(6), object(1)
memory usage: 408.0+ bytes
```

```
bil.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   is_genuine     1500 non-null   bool   
 1   diagonal       1500 non-null   float64
 2   height_left    1500 non-null   float64
 3   height_right   1500 non-null   float64
 4   margin_low     1463 non-null   float64
 5   margin_up      1500 non-null   float64
 6   length         1500 non-null   float64
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

```
bil.loc[bil['margin_low'].isnull(),:]
```

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
72	True	171.94	103.89	103.45	NaN	3.25	112.79
99	True	171.93	104.07	104.18	NaN	3.14	113.08
151	True	172.07	103.80	104.38	NaN	3.02	112.93
197	True	171.45	103.66	103.80	NaN	3.62	113.27
241	True	171.83	104.14	104.06	NaN	3.02	112.36
251	True	171.80	103.26	102.82	NaN	2.95	113.22
284	True	171.92	103.83	103.76	NaN	3.23	113.29
334	True	171.85	103.70	103.96	NaN	3.00	113.36

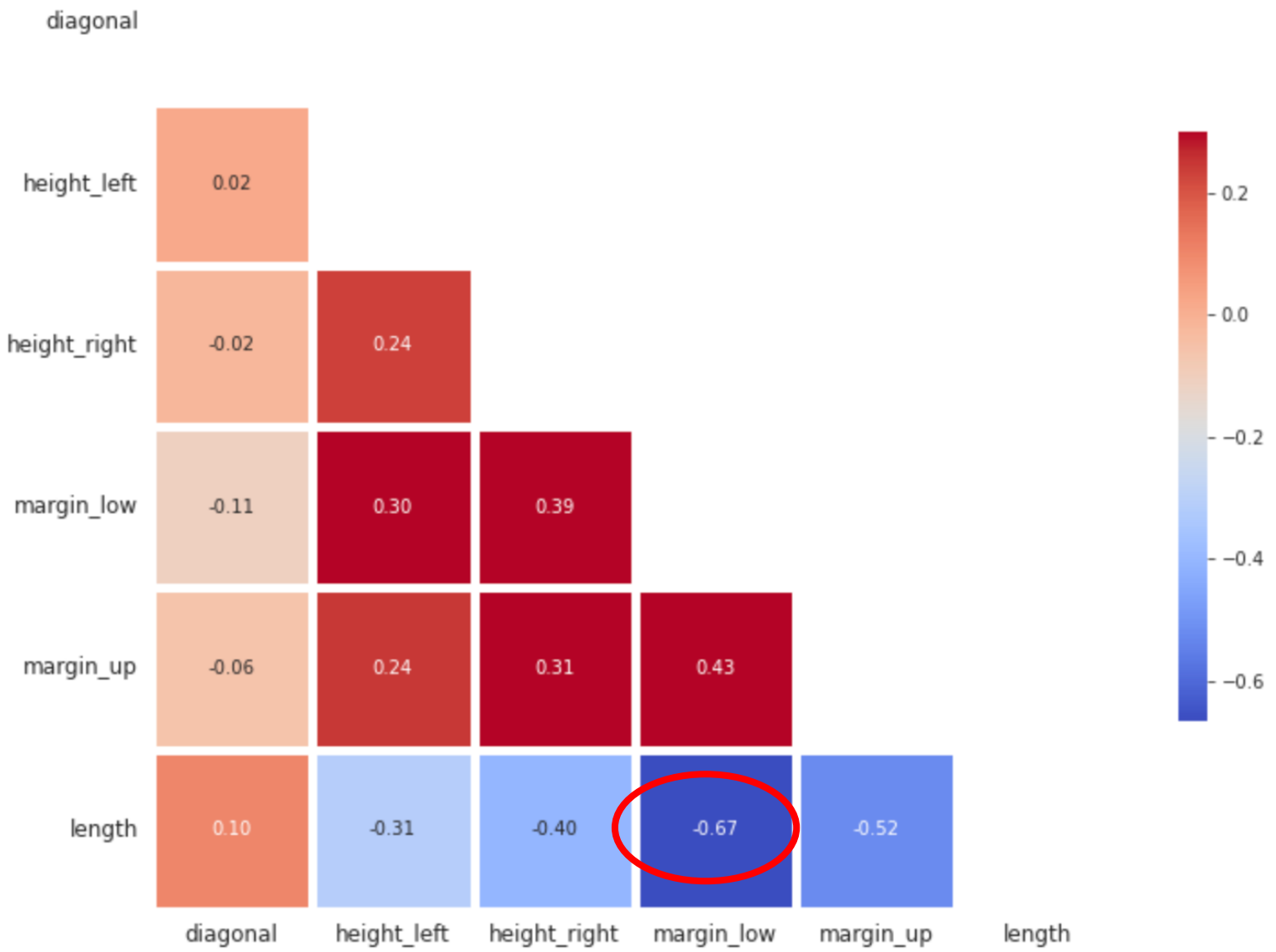
Observation des données

```
# Nous supprimons la colonne 'is_genuine' pour pouvoir effectue
bil_6_df = bil.drop(columns='is_genuine')
```

```
bil_6_df
```

	diagonal	height_left	height_right	margin_low	margin_up	length
0	171.81	104.86	104.95	4.52	2.89	112.83
1	171.46	103.36	103.66	3.77	2.99	113.09
2	172.69	104.48	103.50	4.40	2.94	113.16
3	171.36	103.91	103.94	3.62	3.01	113.51
4	171.73	104.28	103.46	4.04	3.48	112.54

On voit qu'il y a une forte corrélation négative entre 'margin_low' et 'length' avec une valeur de -0.67



Régression linéaire multiple sur le jeu de données

```
reg_multi = smf.ols('margin_low ~ diagonal + height_left + height_right + margin_up + leng
print(reg_multi.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             margin_low   R-squared:                       0.477
Model:                            OLS   Adj. R-squared:                  0.476
Method:                 Least Squares   F-statistic:                     266.1
Date:                Sat, 03 Dec 2022   Prob (F-statistic):          2.60e-202
Time:                        20:13:58   Log-Likelihood:                -1001.3
No. Observations:                1463   AIC:                             2015.
Df Residuals:                    1457   BIC:                             2046.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      22.9948      9.656      2.382      0.017       4.055      41.935
diagonal       -0.1111      0.041     -2.680      0.007      -0.192      -0.030
height_left     0.1841      0.045      4.113      0.000       0.096       0.272
height_right    0.2571      0.043      5.978      0.000       0.173       0.342
margin_up       0.2562      0.064      3.980      0.000       0.130       0.382
length         -0.4091      0.018    -22.627      0.000      -0.445      -0.374
==============================================================================
Omnibus:                       73.627   Durbin-Watson:                   1.893
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               95.862
Skew:                           0.482   Prob(JB):                     1.53e-21
Kurtosis:                       3.801   Cond. No.                     1.94e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifie
d.
[2] The condition number is large, 1.94e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Le R^2 est 0.477, et le R^2 ajusté est d'environ 0.476

-> n'est pas satisfaisant (inférieur à 0,5).

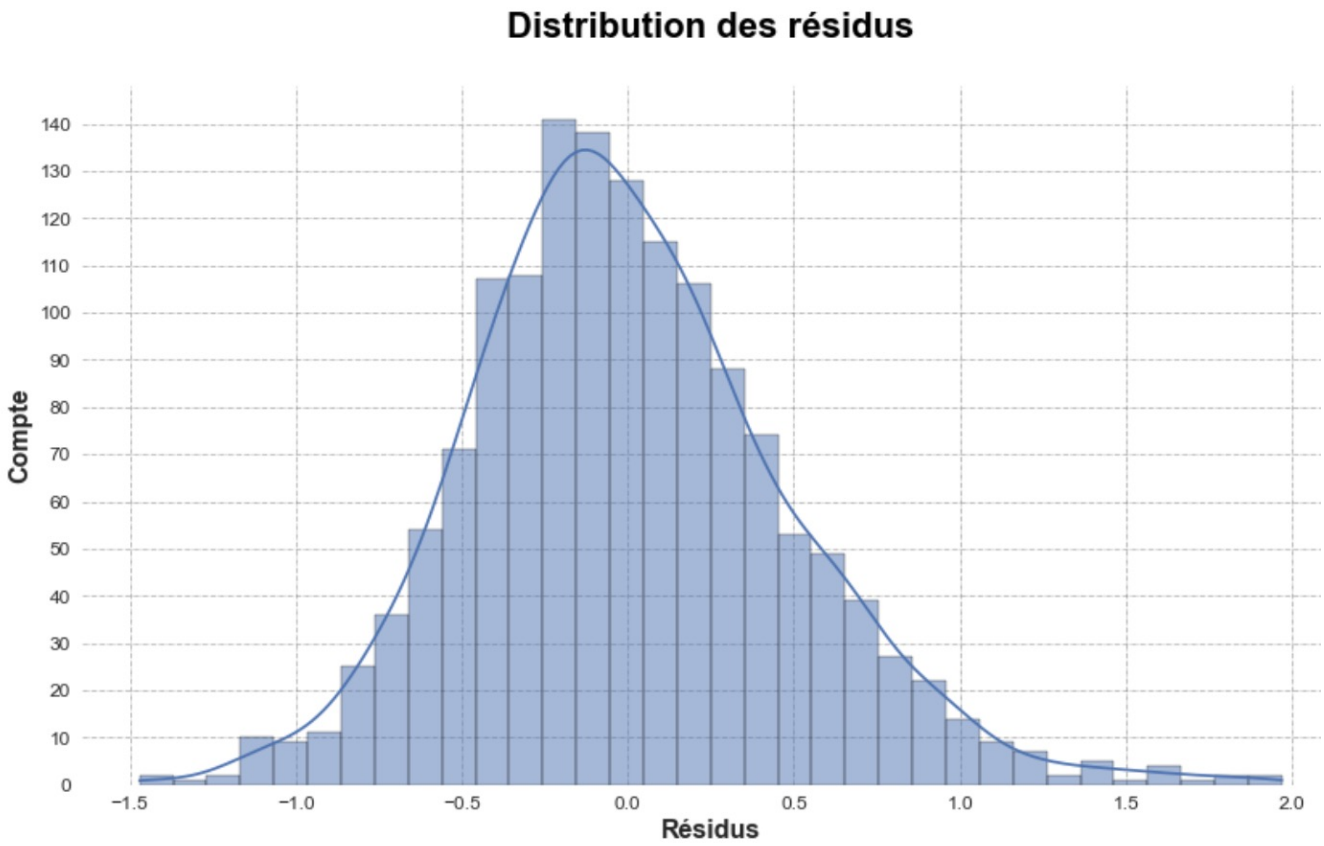
P_value: Ici, elle est très faible et inférieure au seuil alpha 5 %, on rejet l'hypothèse nulle

Prédictation le ‘margin_low’

```
# Nous ajoutons une nouvelle colonne 'margin_low_pred' avec les valeurs prédites de la ré
bil_6_df = bil_6_df.copy()
bil_6_df["margin_low_pred"] = round(reg_multi.predict({'diagonal': bil_6_df['diagonal'],
                                                       'height_left': bil_6_df['height_left'],
                                                       'height_right': bil_6_df['height_right'],
                                                       'margin_up': bil_6_df['margin_up'],
                                                       'length': bil_6_df['length']}),2)
bil_6_df.head()
```

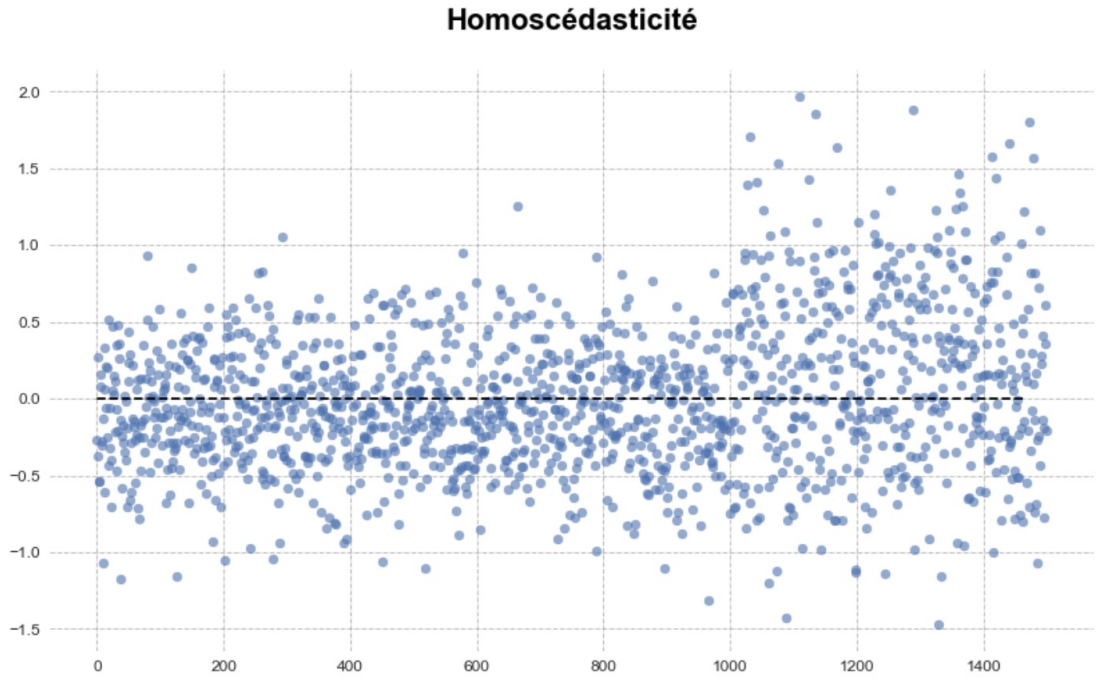
	diagonal	height_left	height_right	margin_low	margin_up	length	margin_low_pred
0	171.81	104.86	104.95	4.52	2.89	112.83	4.79
1	171.46	103.36	103.66	3.77	2.99	113.09	4.14
2	172.69	104.48	103.50	4.40	2.94	113.16	4.13
3	171.36	103.91	103.94	3.62	3.01	113.51	4.16
4	171.73	104.28	103.46	4.04	3.48	112.54	4.58

Normalité de la distribution des résidu



Les résidus suivent une distribution Normale

Test d'homoscédasticité



La p-valeur est inférieure à 5 % -> rejette l'hypothèse $H0$

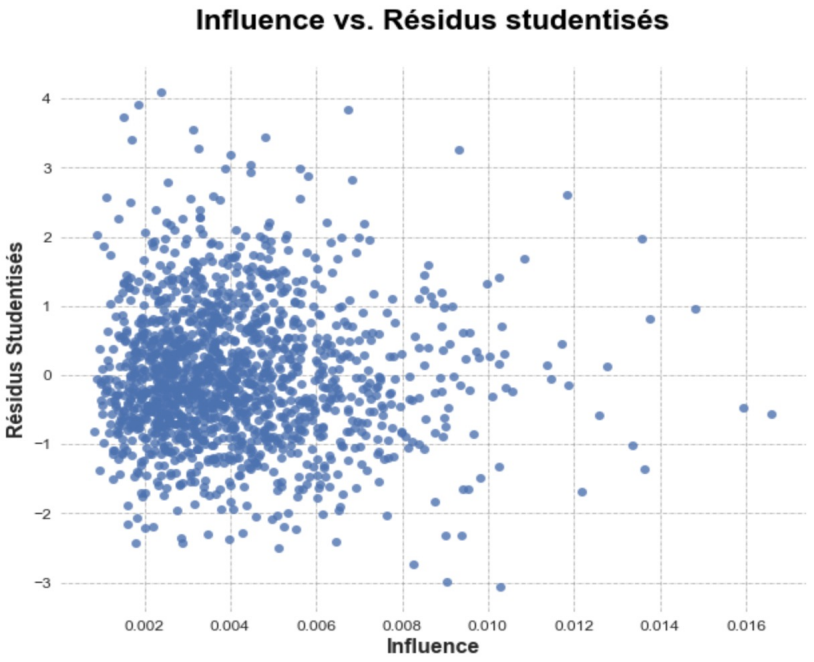
Colinéarité des variables

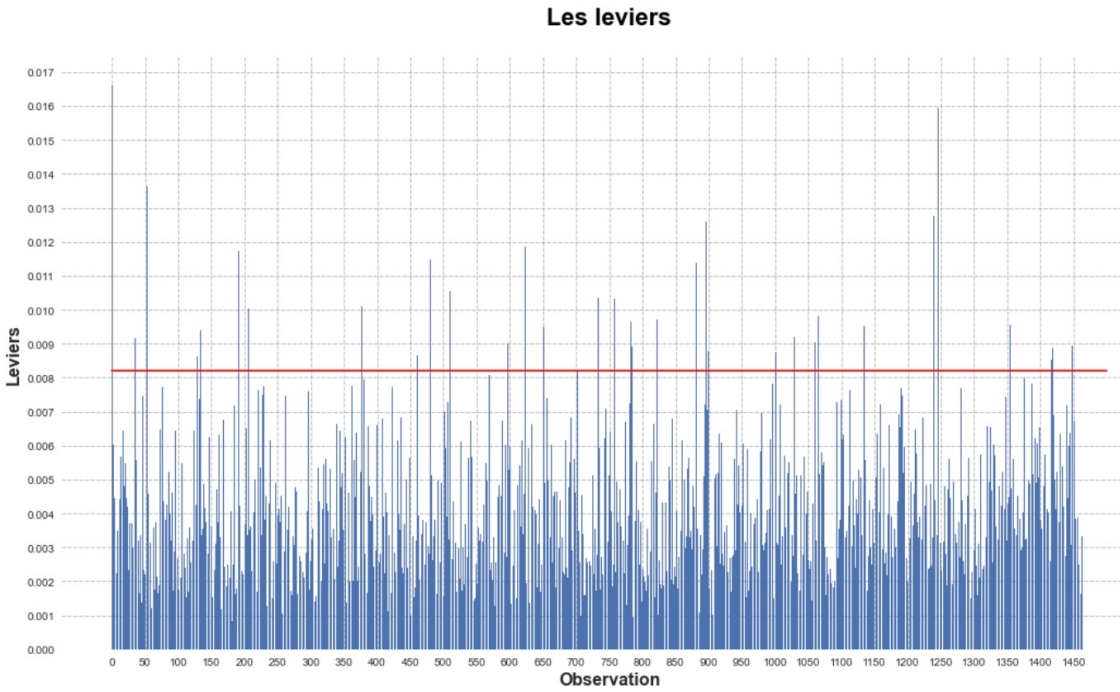
```
variables = reg_multi.model.exog
[variance_inflation_factor(variables, i) for i in np.arange(1,variables.shape[1])]
```

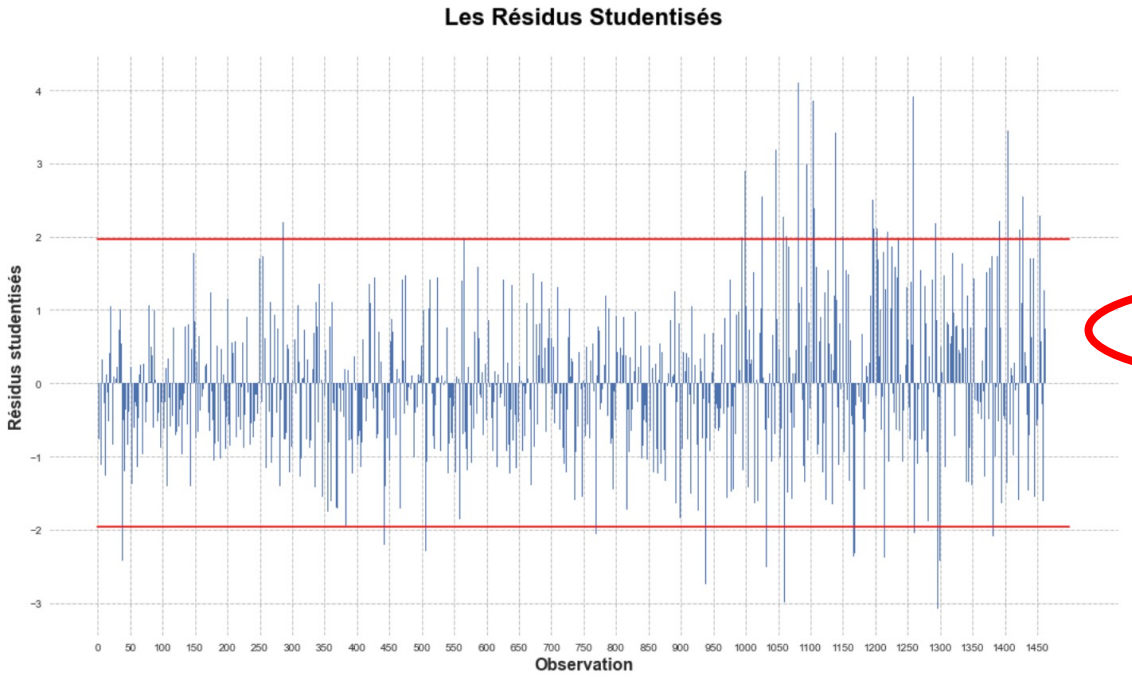
```
[1.0136128990686302,
 1.138260573608248,
 1.2301145155560367,
 1.4044037781843626,
 1.5769501453665806]
```

Pas de problème de colinéarité

Identification...





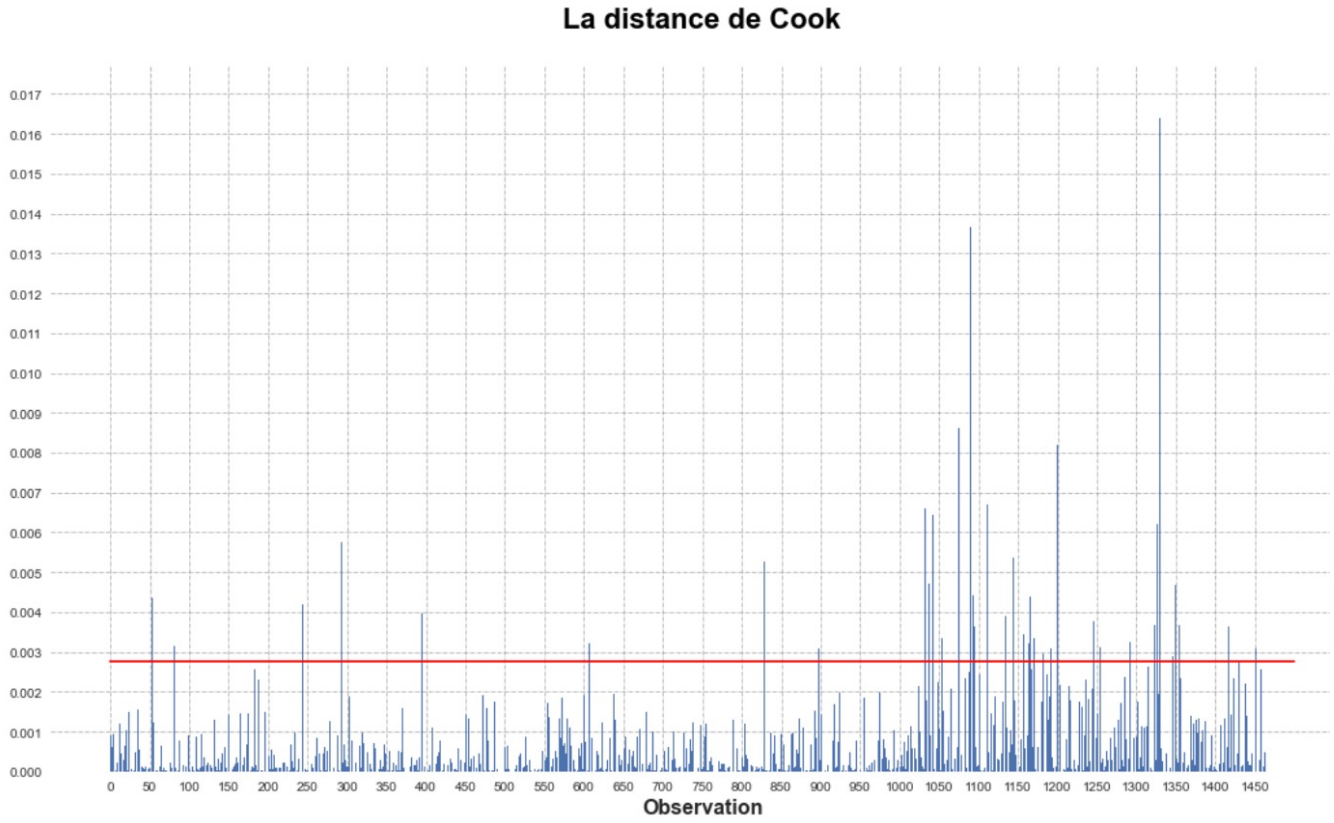



```
# Pour sélectionner les po
dt_n.loc[dt_n['levier'] >
```

	obs	levier
0	1	0.016606
2	3	0.008298
34	35	0.009156
52	53	0.013635
56	57	0.009087
...	...	...
1416	1417	0.008521
1417	1418	0.008870
1428	1429	0.008390
1441	1442	0.009325
1447	1448	0.008931

75 rows × 2 columns

La distance de Cook




```
Résidus studentisés:
 count    1463.000000
mean        0.000133
std         1.001419
min        -3.087577
25%        -0.660578
50%        -0.086814
75%         0.570403
max         4.127099
Name: student_resid, dtype: float64
Nombre outliers: margin_low    79
dtype: int64
```

```
count    1463.000000
mean        0.004101
std         0.002134
min         0.000840
25%         0.002561
50%         0.003645
75%         0.005140
max         0.016606
Name: leverage, dtype: float64
Nombre d'individus à forte influence: 75
```

```
margin_low_x    8
margin_low_y    8
dtype: int64
```

Identification les valeur atypique

	margin_low_x	margin_low_y
664	5.04	5.04
965	3.45	3.45
1022	6.03	6.03
1074	4.36	4.36
1089	3.86	3.86
1199	4.49	4.49
1329	3.82	3.82
1478	6.08	6.08

Relance la régression linéaire multiple

OLS Regression Results

Dep. Variable:	margin_low	R-squared:	0.493
Model:	OLS	Adj. R-squared:	0.491
Method:	Least Squares	F-statistic:	281.5
Date:	Sat, 03 Dec 2022	Prob (F-statistic):	1.45e-210
Time:	20:14:11	Log-Likelihood:	-969.98
No. Observations:	1455	AIC:	1952.
Df Residuals:	1449	BIC:	1984.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	24.9633	9.511	2.625	0.009	6.306	43.620
diagonal	-0.1277	0.041	-3.106	0.002	-0.208	-0.047
height_left	0.1981	0.044	4.489	0.000	0.112	0.285
height_right	0.2544	0.042	5.997	0.000	0.171	0.338
margin_up	0.2791	0.064	4.356	0.000	0.153	0.405
length	-0.4122	0.018	-22.992	0.000	-0.447	-0.377

Omnibus:	75.350	Durbin-Watson:	1.909
Prob(Omnibus):	0.000	Jarque-Bera (JB):	94.555
Skew:	0.508	Prob(JB):	2.93e-21
Kurtosis:	3.727	Cond. No.	1.94e+05

Application du modèle

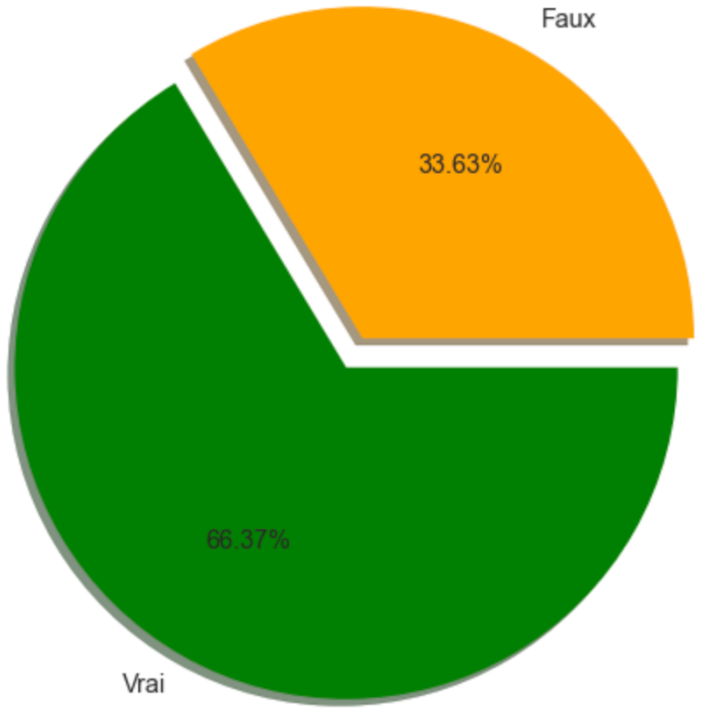
	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	1	171.81	104.86	104.95	4.52	2.89	112.83
1	1	171.46	103.36	103.66	3.77	2.99	113.09
2	1	172.69	104.48	103.50	4.40	2.94	113.16
3	1	171.36	103.91	103.94	3.62	3.01	113.51
4	1	171.73	104.28	103.46	4.04	3.48	112.54

```
# describe
df_final.describe()
```

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
count	1463.000000	1463.000000	1463.000000	1463.000000	1463.000000	1463.000000	1463.000000
mean	0.663705	171.959193	104.031333	103.921476	4.485967	3.153083	112.674757
std	0.472603	0.305457	0.299605	0.324181	0.663813	0.231466	0.873222
min	0.000000	171.040000	103.140000	102.910000	2.980000	2.270000	109.490000
25%	0.000000	171.750000	103.825000	103.710000	4.015000	2.990000	112.020000
50%	1.000000	171.960000	104.040000	103.920000	4.310000	3.140000	112.960000
75%	1.000000	172.170000	104.230000	104.150000	4.870000	3.315000	113.340000
max	1.000000	173.010000	104.880000	104.950000	6.900000	3.910000	114.320000

Analyse descriptive

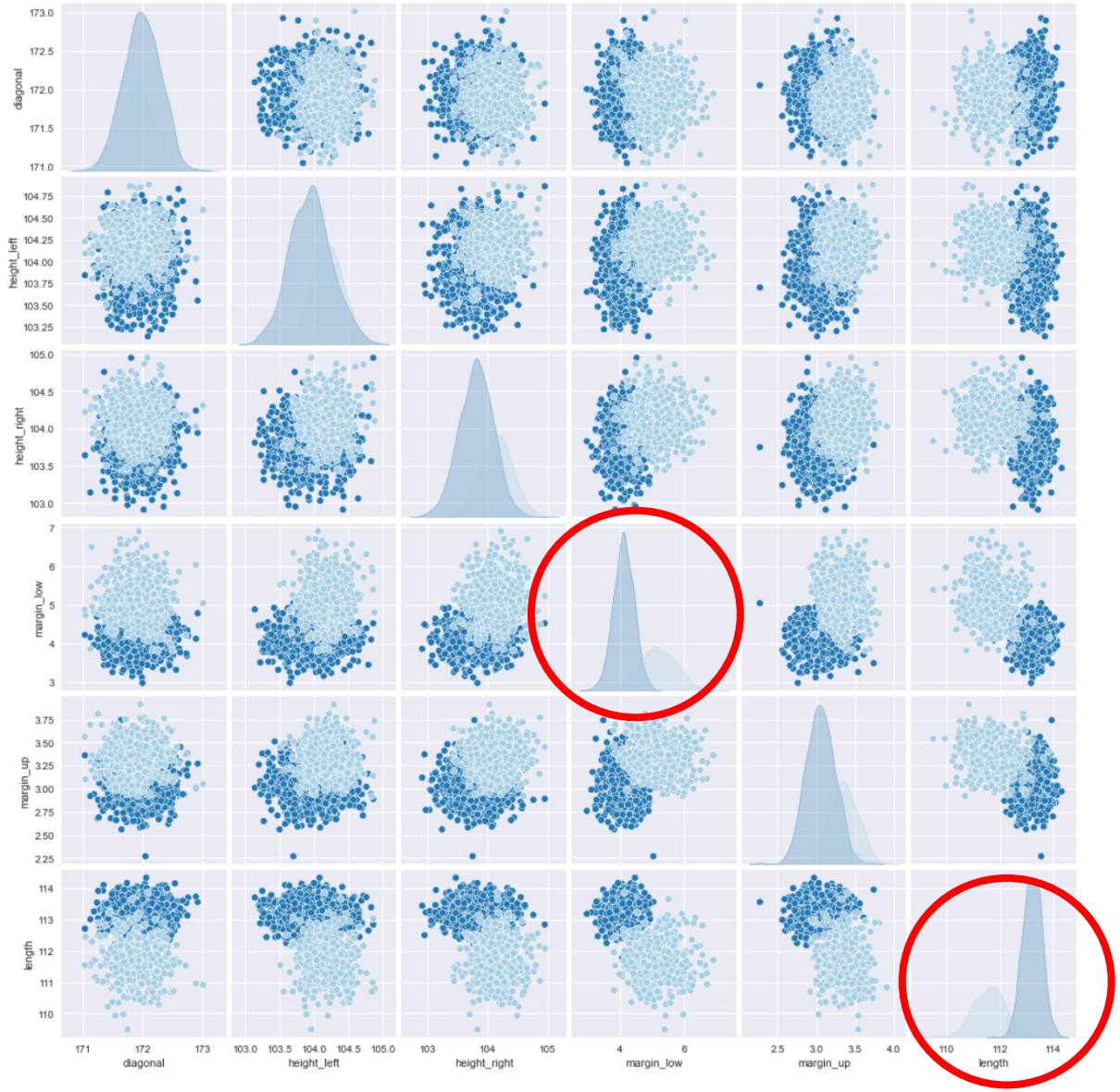
is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	492	492	492	492	492	492
1	971	971	971	971	971	971

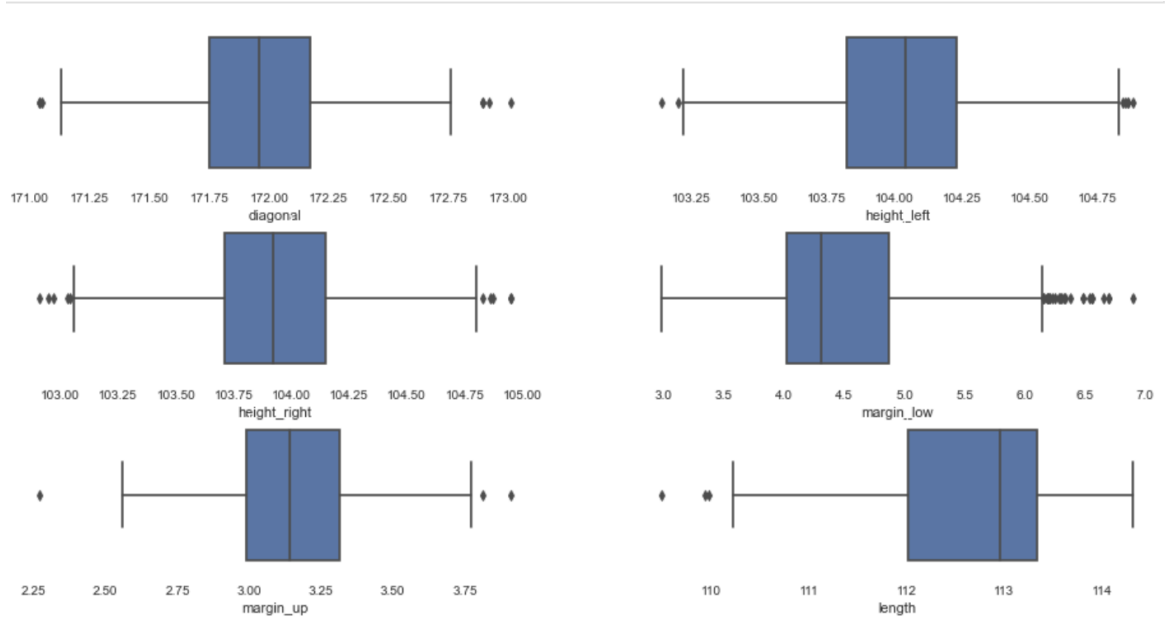



```
print(df_final.groupby(["is_genuine"]).mean())
```

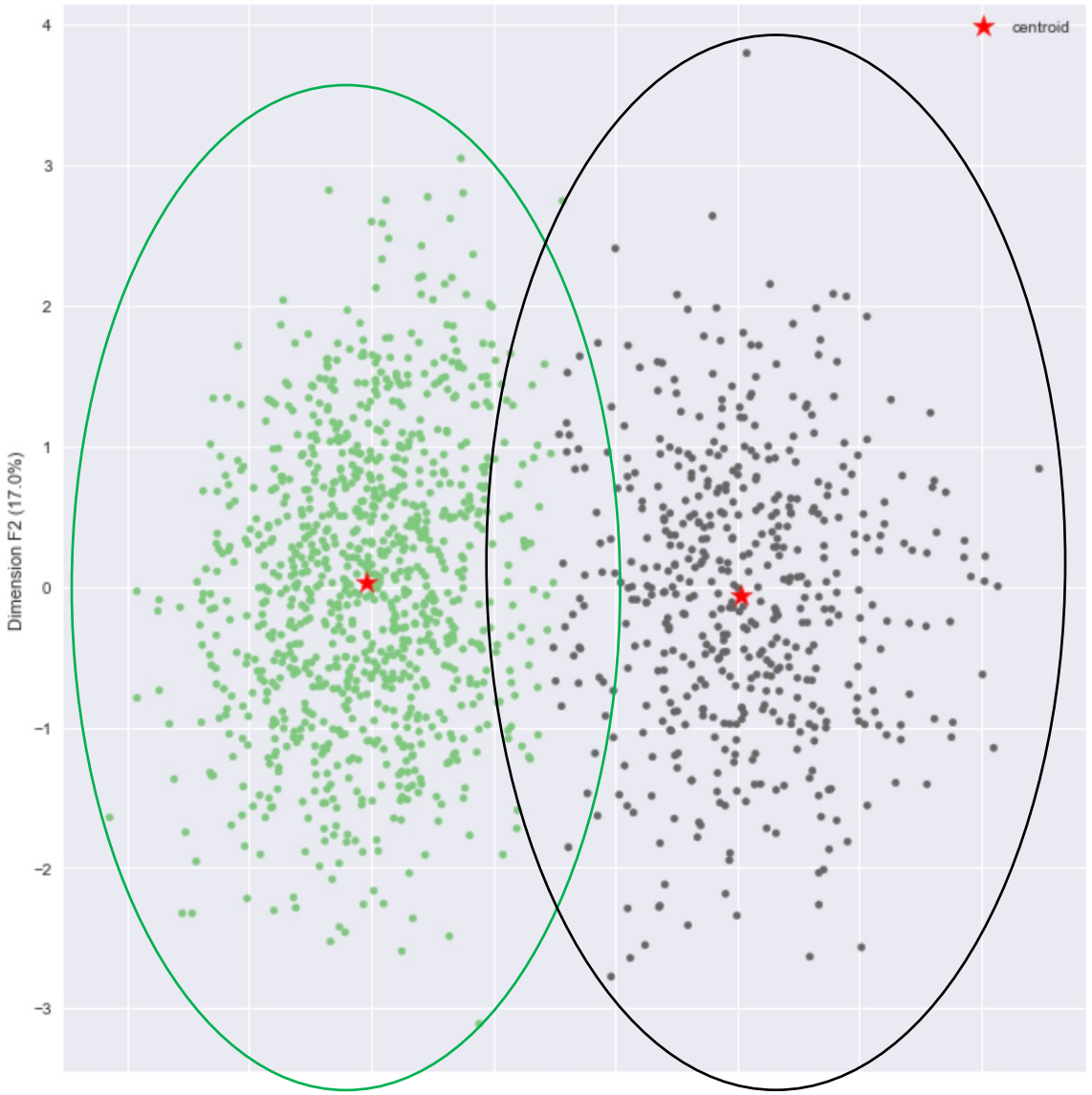
```
              diagonal  height_left  height_right  margin_low  margin_up  \
is_genuine
0           171.901402   104.188537    104.143272    5.215935   3.351504
1           171.988476   103.951679    103.809094    4.116097   3.052544

                length
is_genuine
0           111.632114
1           113.203059
```

Pair Plot et Box plot

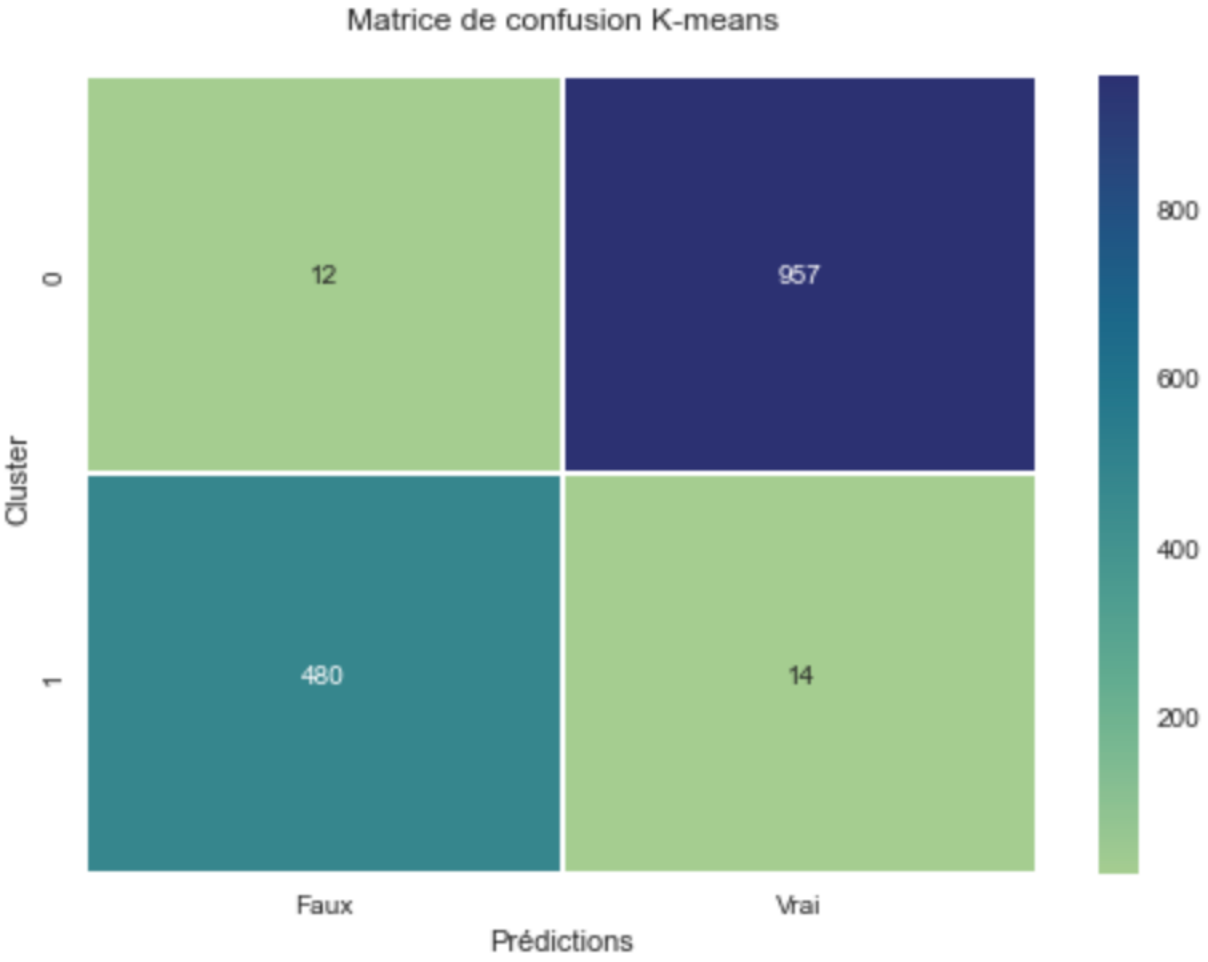




K-means et ACP



Matrice de Confusion



Cluster 0: Vrais billets
Cluster 1: Faux billets

Prédiction K-means

```
# Predictions sur des donnees inconnues:
x_test_km = billets_prod[["diagonal","height_left","height_right","margin_low","margin_up'

billets_prod["cluster_pred"] = kmeans2.predict(x_test_km)
print(billets_prod[["id","cluster_pred"]])
```

```
    id  cluster_pred
0  A_1             0
1  A_2             0
2  A_3             0
3  A_4             1
4  A_5             1
```

Les 3 premiers billets appartiennent au cluster 0, qui sont le groupe des faux billets, et les 2 derniers appartiennent au cluster 1, des vrais billets!!

Régression logistique

```
Optimization terminated successfully.
         Current function value: 0.026765
         Iterations 12
                           Logit Regression Results
==============================================================================
Dep. Variable:              is_genuine   No. Observations:                 1463
Model:                          Logit   Df Residuals:                     1456
Method:                           MLE   Df Model:                            6
Date:                Sat, 03 Dec 2022   Pseudo R-squ.:                  0.9581
Time:                        20:14:17   Log-Likelihood:                -39.158
converged:                       True   LL-Null:                       -934.20
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       -243.4159    249.370     -0.976      0.329    -732.172     245.340
diagonal       0.4029      1.159      0.348      0.728      -1.869       2.675
height_left   -1.1602      1.125     -1.032      0.302      -3.365       1.044
height_right  -3.1087      1.224     -2.540      0.011      -5.508      -0.710
margin_low    -5.9078      0.983     -6.010      0.000      -7.834      -3.981
margin_up    -10.2367      2.199     -4.656      0.000     -14.546      -5.927
length         6.0242      0.893      6.750      0.000       4.275       7.773
==============================================================================
```

-> recommence la régression sans ces variables (diagonal, height_left).

Régression logistique des variables significatives

- P-value est inférieure à 0.05, le modèle est significatif.

- Le R^2 est très proche de 1,
-> la majorité de la variance observée.

```
Optimization terminated successfully.
         Current function value: 0.027189
         Iterations 13
                         Results: Logit
=================================================================
Model:                Logit            Pseudo R-squared: 0.957
Dependent Variable:   is_genuine       AIC:              89.5549
Date:                 2022-12-03 20:14 BIC:              115.9961
No. Observations:     1463             Log-Likelihood:   -39.777
Df Model:             4                LL-Null:          -934.20
Df Residuals:         1458             LLR p-value:      0.0000
Converged:            1.0000           Scale:            1.0000
No. Iterations:       13.0000
-----------------------------------------------------------------
                Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
-----------------------------------------------------------------
const        -258.7502 143.3183 -1.8054 0.0710 -539.6488 22.1485
height_right   -3.5542   1.1885 -2.9904 0.0028   -5.8837 -1.2247
margin_low     -6.1958   0.9590 -6.4607 0.0000   -8.0754 -4.3162
margin_up     -10.3632   2.1886 -4.7350 0.0000  -14.6529 -6.0735
length          6.1300   0.8862  6.9174 0.0000    4.3931  7.8669
=================================================================
```

Test binarité

```
print(df_final["is_genuine"].nunique())
```

```
2
```

2 valeurs possibles pour la variable prédite

Multicollinératité

```
vif= pd.Series([variance_inflation_factor(X_billet.values, i)
                for i in range(X_billet.shape[1])],
               index=X_billet.columns)
print("VIF par features:\n", vif)
print("VIF moyen:",vif.mean())
```

```
VIF par features:
 const            187159.848539
height_right          1.247632
margin_low            1.883516
margin_up             1.412027
length                2.103478
dtype: float64
VIF moyen: 37433.29903837327
```

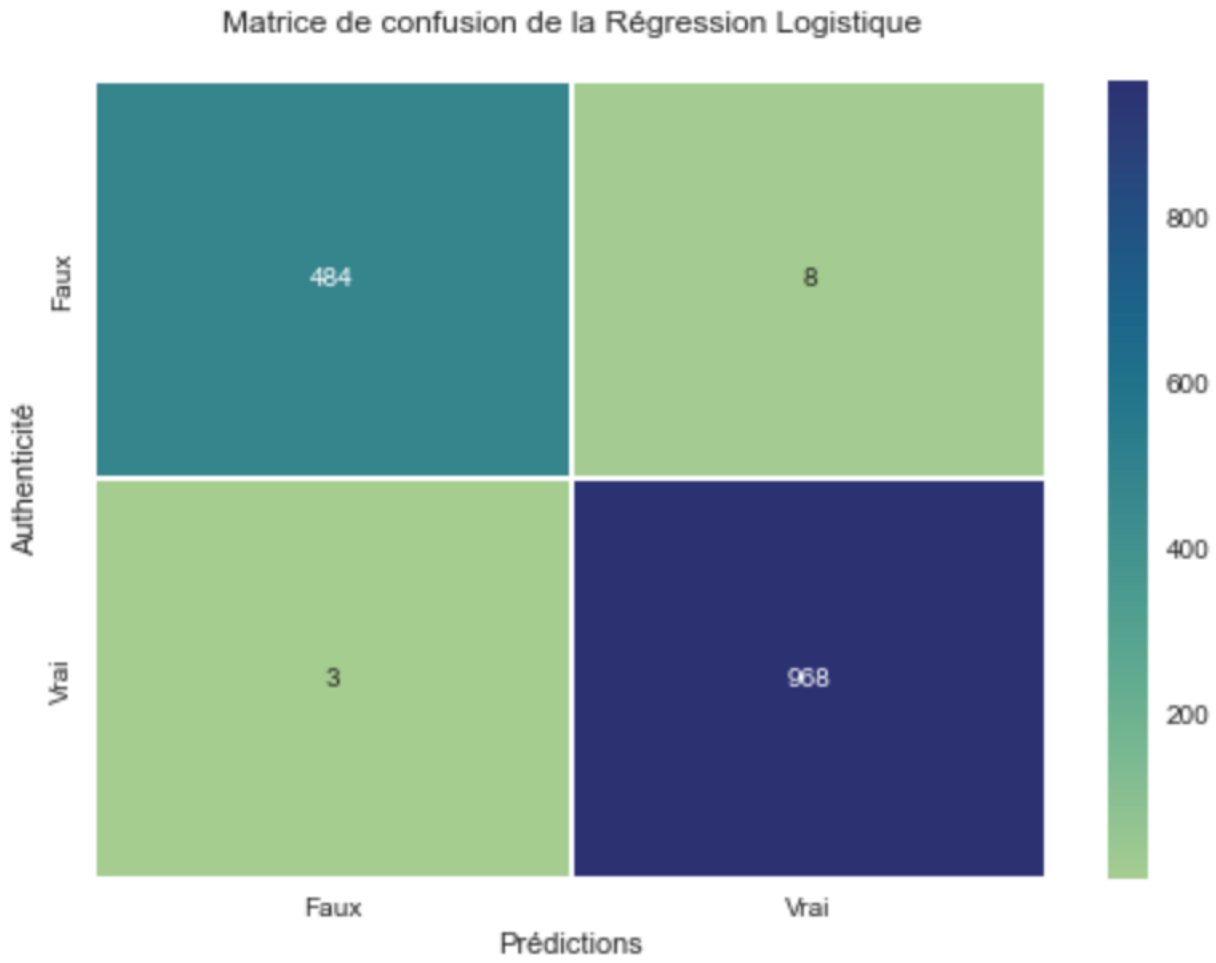
pas de collinéarité

Application du modèle

```
# Application du modelé sur les données training:
df_final["proba"] = model_reg_log.predict(X_billet)
df_final["y_pred"] = (model_reg_log.predict(X_billet) >= 0.5).astype(int)
df_final.head()
```

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length	proba	y_pred
0	1	171.81	104.86	104.95	4.52	2.89	112.83	0.873599	1
1	1	171.46	103.36	103.66	3.77	2.99	113.09	0.999992	1
2	1	172.69	104.48	103.50	4.40	2.94	113.16	0.999912	1
3	1	171.36	103.91	103.94	3.62	3.01	113.51	0.999999	1
4	1	171.73	104.28	103.46	4.04	3.48	112.54	0.909772	1

```
[[32.26666667  0.53333333]
 [ 0.2        64.53333333]]
```



Score de classification de précision

```
Accuracy Score:  0.9924812030075187
Precision Score:  0.9918032786885246
Recall Score:  0.9969104016477858
Score F1:  0.9943502824858756
```

```
print(classification_report(df_final["is_genuine"], df_final["y_pred"]))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       492
           1       0.99      1.00      0.99       971

    accuracy                           0.99      1463
   macro avg       0.99      0.99      0.99      1463
weighted avg       0.99      0.99      0.99      1463
```

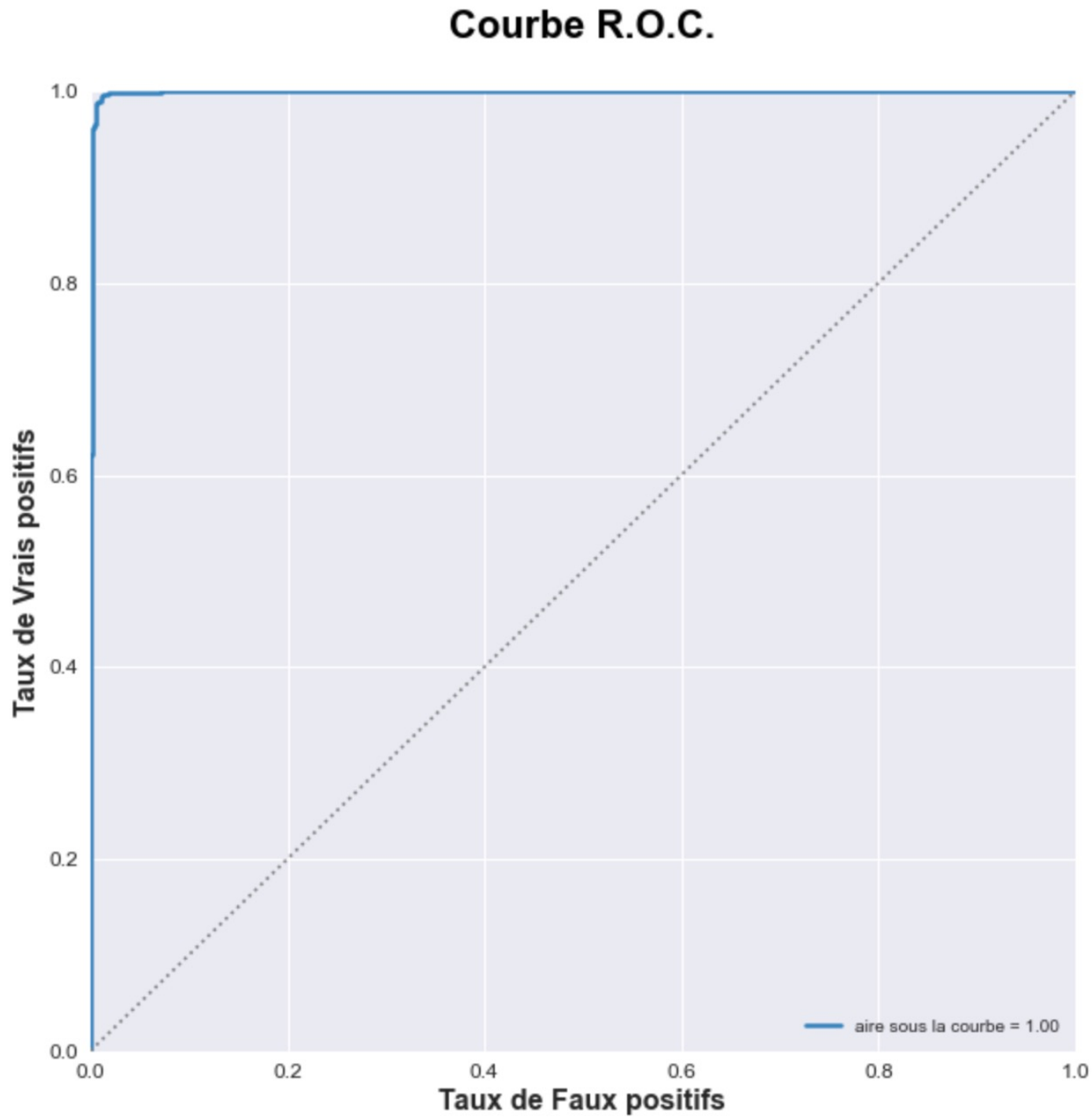
tous les scores s'approchent de 1!!!!

Courbe R.O.C

```
print("AUC :",roc_auc)
```

```
AUC : 0.9989240829586462
```

AUC = 0.9989
-> une aire sous la courbe approximée à 1.



Application du modèle aux données test

```
    id      proba  y_pred
0  A_1   0.000036       0
1  A_2   0.000002       0
2  A_3   0.000010       0
3  A_4   0.995567       1
4  A_5   0.999992       1

Identification des billets:
Le billet A_1 est faux
Le billet A_2 est faux
Le billet A_3 est faux
Le billet A_4 est vrai
Le billet A_5 est vrai
```

billets_prod

	diagonal	height_left	height_right	margin_low	margin_up	length	id	cluster_pred	proba	y_pred
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1	0	0.000036	0
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2	0	0.000002	0
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3	0	0.000010	0
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4	1	0.995567	1
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5	1	0.999992	1

Algorithme de prédiction de billets

```
evaluation('billets_production.csv')
```

```
On va estimer les billets du fichier: billets_production.csv
    id     proba  y_pred
0  A_1  0.000036       0
1  A_2  0.000002       0
2  A_3  0.000010       0
3  A_4  0.995567       1
4  A_5  0.999992       1

Identification des billets:
Le billet A_1 est faux
Le billet A_2 est faux
Le billet A_3 est faux
Le billet A_4 est vrai
Le billet A_5 est vrai
```

Conclusion

- statistique descriptive
- analyse en composantes principales
- classification automatique
- modélisation de type régression logistique

Détectez des faux billets

Merci pour votre attention