Projet 6_Détectez des faux billets

Parcours Data Analyst

Xuefei ZHANG_janvier 2022

Sommaire

- Analyse descriptive
- ACP
- Classification - K means
- Modélisation - régression logistique
- Cas pratique: test de programme

M0 - Analyse de statistique descriptive

M0 - affichage et nettoyage du jeu de données

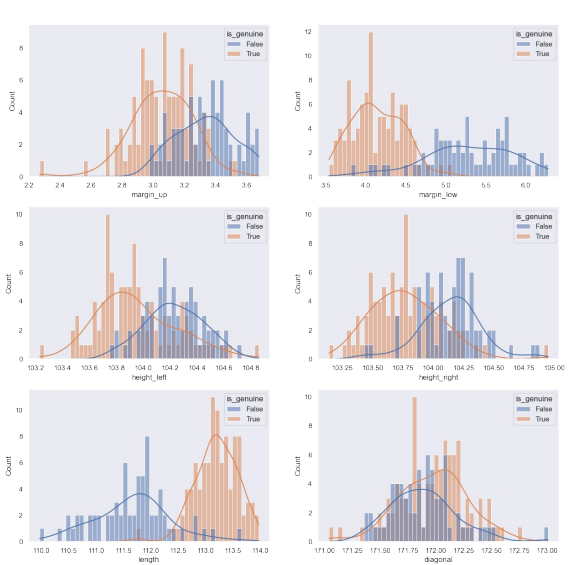
```
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   is_genuine    170 non-null    bool
 1   diagonal      170 non-null    float64
 2   height_left   170 non-null    float64
 3   height_right  170 non-null    float64
 4   margin_low    170 non-null    float64
 5   margin_up     170 non-null    float64
 6   length        170 non-null    float64
dtypes: bool(1), float64(6)
memory usage: 8.3 KB
None
Index(['is_genuine', 'diagonal', 'height_left', 'height_right', 'margin_low',
       'margin_up', 'length'],
      dtype='object')
```

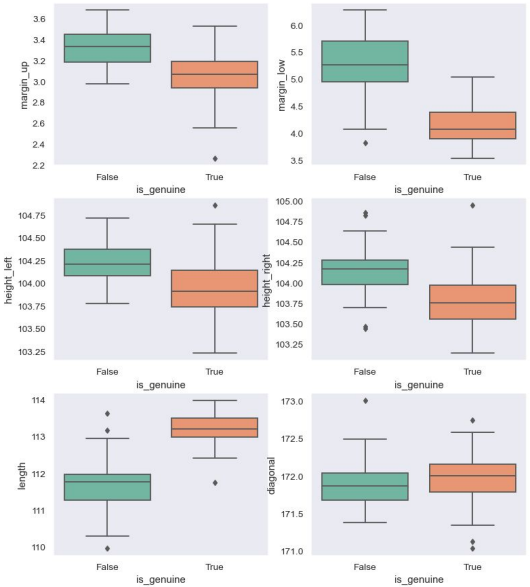
Out[20]:

	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	171.67	103.74	103.70	4.01	2.87	113.29
2	True	171.83	103.76	103.76	4.40	2.88	113.84
3	True	171.80	103.78	103.65	3.73	3.12	113.63
4	True	172.05	103.70	103.75	5.04	2.27	113.55

Jeu de données bien propre pour l'analyse

M0 - Analyse univariée





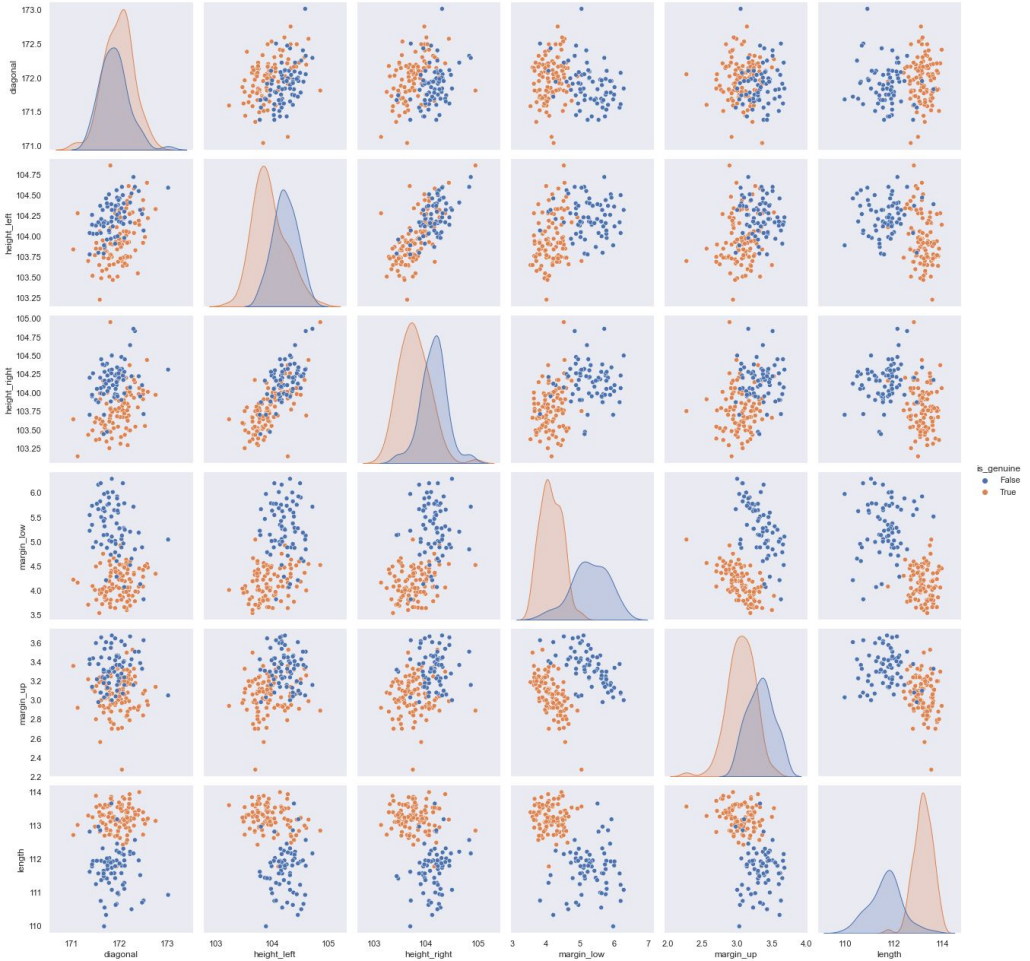

```
notesO.groupby(by='is_genuine').mean().transpose()
```

is_genuine	False	True
diagonal	171.889857	171.9761
height_left	104.230429	103.9515
height_right	104.145571	103.7759
margin_low	5.281571	4.1435
margin_up	3.334571	3.0555
length	111.660714	113.2072

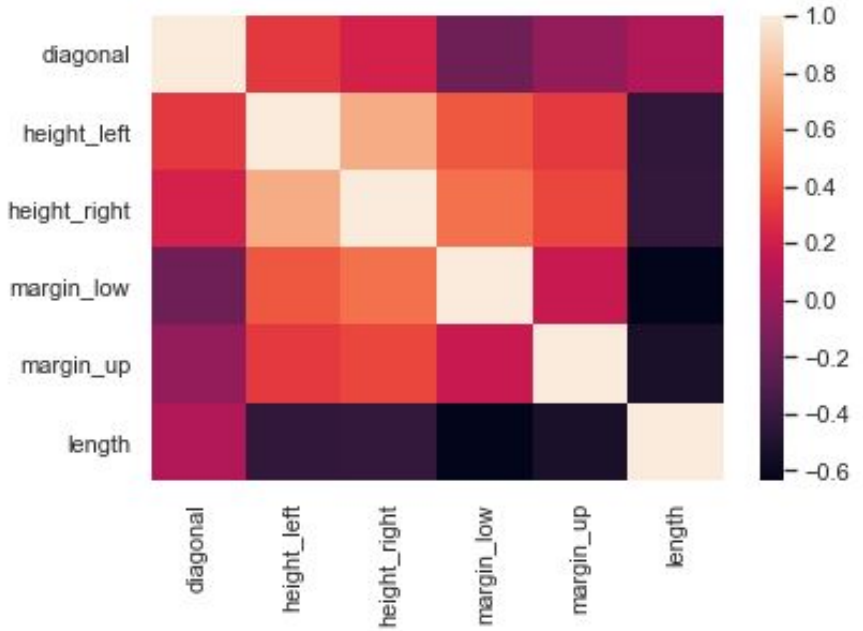
À noter que:
il existe des différences remarquables entre les données de vrais billets et faux billets sur les 5 variables sauf “diagonal” , en terme de : **distribution de densité, médiane, moyenne**

M0 - Analyse bivariée

Les individus True et False de classification is_genuine se regroupent en 2 partitions sur tous ces 6 variables (mesures)



M0 - Analyse bivariée

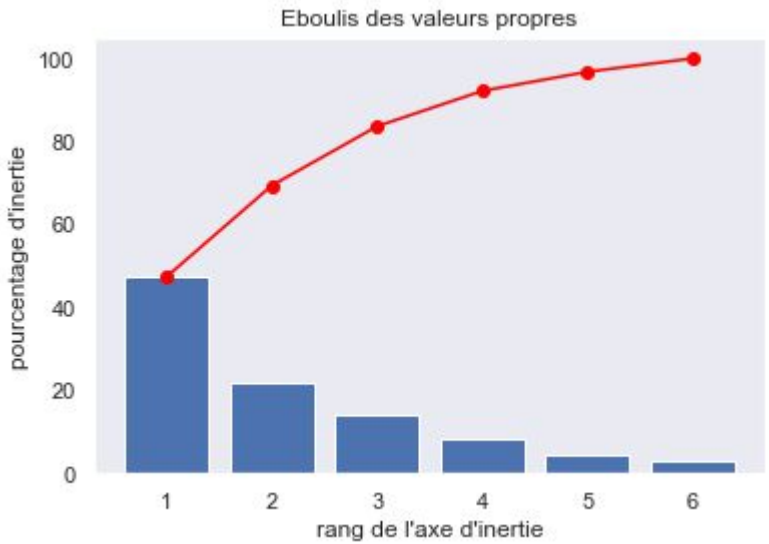


Corrélation entre les variables:

- height_left vs height_right : corrélation 70%
- margin_low vs height_right : 50%
- margin_up vs heights : 30%
- length vs heights: - 50%
- margin_low vs length : - 60%

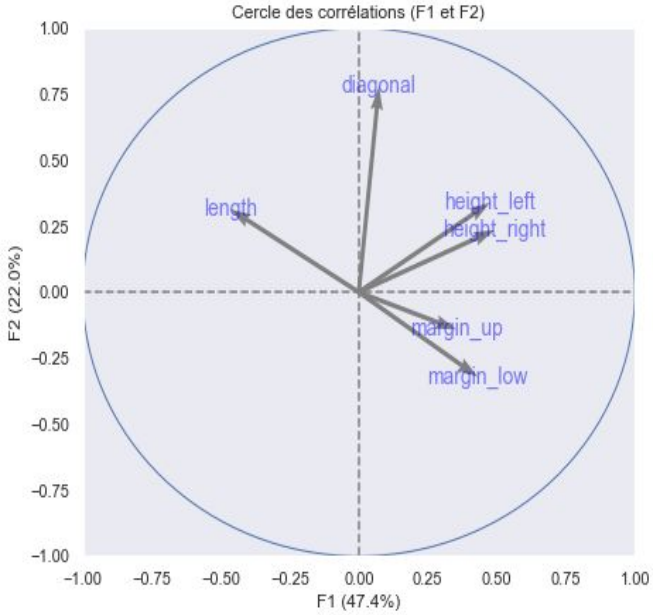
M1 - Analyse de Composantes Principales (PCA)

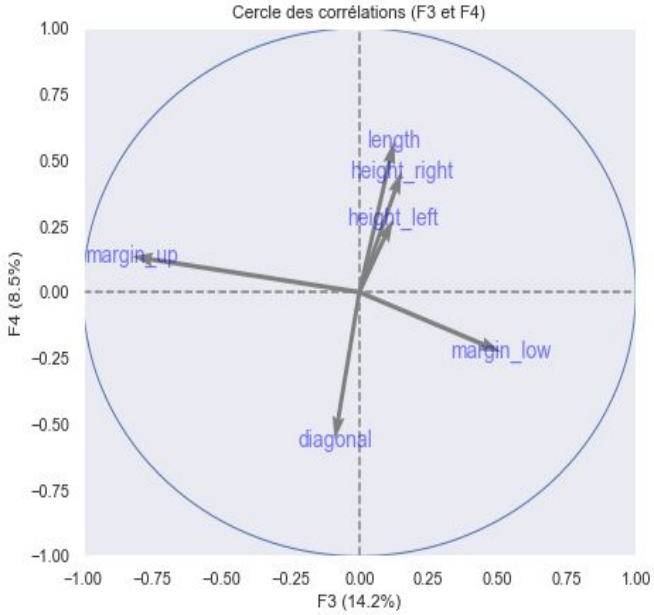
ACP - Eboulis de valeurs propres



- Les 2 premiers facteurs (le 1er plan factoriel) représentent environ **70%** de l'inertie
- Les 4 premiers facteurs (les 2 premiers plans factoriels) représentent environ **90%** de l'inertie

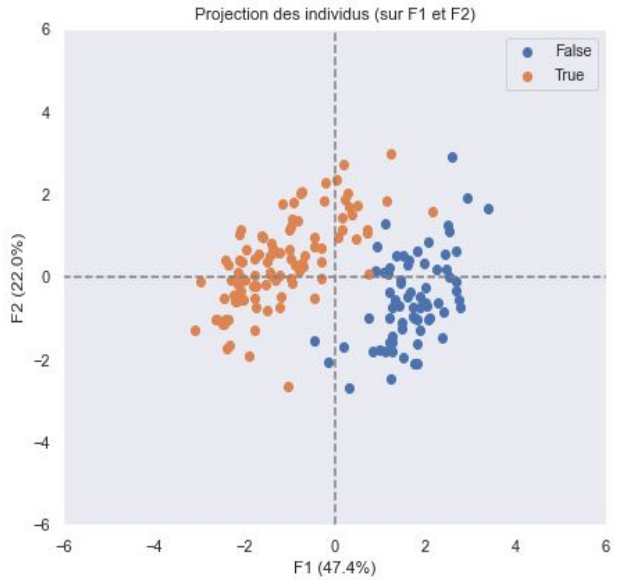
ACP - représentation des variables par cercle des corrélations

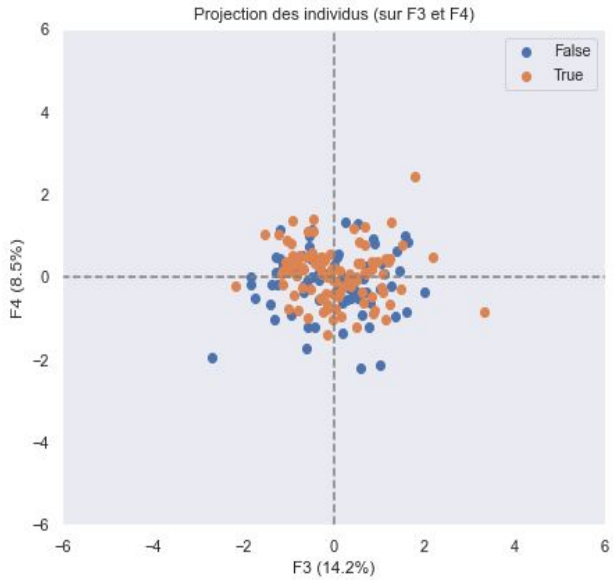




- le **1er plan** représente globalement bien les 6 variables
- **diagonal** est quasiment superposé avec F2 au sens positif, et longue flèche => bien représenté par 1er plan factoriel
- sur le 1er et 2ème plan, **height_left et height_right** se corrèlent bien vu l'angle entre eux
- **margin_up** flèche longue et petit angle avec F3 au sens négatif => bien représenté par F3 au sens négatif

ACP - Projection des individus par 3 plans factoriels - nuage de points





* variable illustrative: **is_genuine** (**False**/**True**)

- Les individus False et True se distinguent et se groupent en 2 clusters sur le 1er plan factoriel
- Sur 2ème et 3ème plans, il n'y a pas de clusters clairement formés

ACP - Qualité de représentation des individus

In [214]:

```
cos2 = coord**2
print(cos2.shape)

for j in range(p):
    cos2[:,j] = cos2[:,j]/di

repres= pd.DataFrame({'genuine':notesO.index,'COS2_1':cos2[:,0],'COS2_2':cos2[:,1]})
repres

#comme ça, Les COS² pour les 2 premiers facteurs sont affichés
```

(170, 6)

Out[214]:

	genuine	COS2_1	COS2_2
0	0	0.251929	0.139000
1	1	0.818002	0.050822
2	2	0.784862	0.000466
3	3	0.882856	0.001652
4	4	0.320145	0.009417
...	...	...	...
165	165	0.800651	0.004703
166	166	0.324059	0.411824
167	167	0.498809	0.083461
168	168	0.156908	0.271800
169	169	0.421817	0.217111

170 rows × 3 columns

*COS2_1, COS2_2: Qualité de représentation des 170 individus sur les 2 premiers facteurs (1er plan factoriel)

Conformément à la théorie de projection, pour chaque individu, la **somme des COS² sur l'ensemble des 6 facteurs** est égale à **1**.

```
print(np.sum(cos2,axis=1))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1.]
```

ACP - Contribution des individus aux facteurs

ctr: permet de déterminer les individus qui pèsent le plus dans la définition de chaque facteur.

In [216]:
```
#contributions aux axes
ctr = coord**2
for j in range(p):
    ctr[:,j] = ctr[:,j]/(n*eigval[j])

ctr_axes= pd.DataFrame({'genuine': notesO.index,'CTR_1':ctr[:,0],'CTR_2':ctr[:,1]})
# ici on prends axe 1 et 2

ctr_axes.sort_values(by="CTR_1", ascending= False ).head()
# les 5 individus qui contributent le plus aux CTR_1
```

Out[216]:

	genuine	CTR_1	CTR_2
122	122	0.023758	0.012372
49	49	0.019620	0.007487
29	29	0.018089	0.000038
112	112	0.017950	0.016259
158	158	0.015836	0.002423

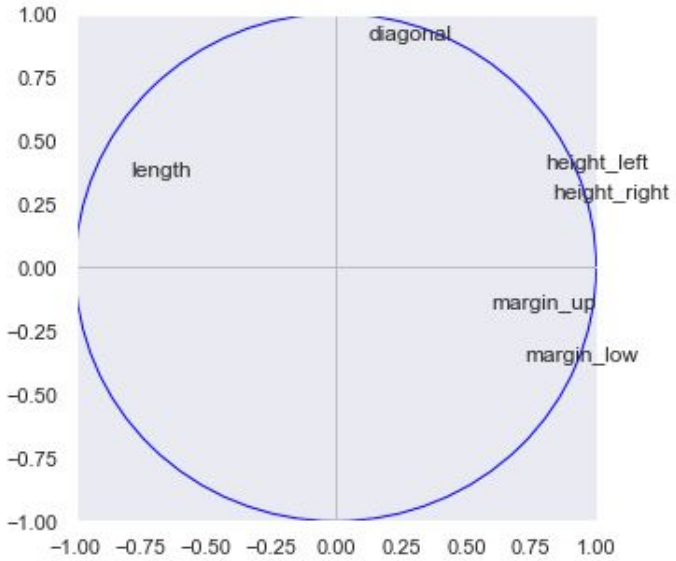
Conformément à la théorie, pour chaque axe, la **somme des contributions des individus sur cette axe** est égale à **1**.

In [217]:
```
#vérifions la théorie
print(np.sum(ctr,axis=0))
```

```
[1. 1. 1. 1. 1. 1.]
```

ACP - corrélations des 6 variables avec F1 et F2 (1er plan factoriel)

	id	COR_1	COR_2
0	diagonal	0.123635	0.894863
1	height_left	0.802300	0.389389
2	height_right	0.829835	0.270354
3	margin_low	0.727258	-0.367910
4	margin_up	0.594829	-0.161976
5	length	-0.785209	0.361022



Constatations:

- **4 variables (height_left, height_right, margin_up, margin_low)** sont positivement corrélées avec F1 avec des coeff de corrélation élevés (>0.6);
- **diagonal** n'est pas bien corrélé avec F1, mais bien corrélé avec F2 au sens positif (0.89);
- **length** négativement corrélé avec F1 à un coeff - 0.78

ACP - Qualité de représentation des variables (COS²)

In [215]:
```
#cosinus carré des variables sur F1 et F2
cos2var = corvar**2
print(cos2var.shape)
repres_variables = pd.DataFrame({'id':notes.columns,'COS2_1':cos2var[:,0],'COS2_2':cos2var[:,1]})
repres_variables['COS2_1+2'] = repres_variables['COS2_1'] + repres_variables['COS2_2']
repres_variables
```

```
(6, 6)
```

Out[215]:

	id	COS2_1	COS2_2	COS2_1+2
0	diagonal	0.015286	0.800781	0.816066
1	height_left	0.643685	0.151624	0.795308
2	height_right	0.688626	0.073091	0.761717
3	margin_low	0.528904	0.135358	0.664262
4	margin_up	0.353822	0.026236	0.380058
5	length	0.616553	0.130337	0.746890

Constatation:

- F1 représente bien 5 variables sauf diagonal
- F2 représente bien diagonal
- F1+F2 (COS2_1+2) représente dans son ensemble bien ces 6 variables

In [224]:
```
#verification qualité repésentation des variables sur facteurs
print(np.sum(cos2var,axis=1))
```

```
[1. 1. 1. 1. 1. 1.]
```

La somme des COS² sur toutes les composantes sont égales à 1 (la **somme des COS² d'une variable sur l'ensemble des 6 facteurs** est égale à **1**)

ACP - Contribution des variables aux axes (CTR)

In [128]:
```
#contributions
ctrvar = cos2var

for k in range(p):
    ctrvar[:,k] = ctrvar[:,k]/eigval[k]

# ici on n'affiche que pour les deux premiers axes
contri_variables= pd.DataFrame({'id':notes.columns,'CTR_F1':ctrvar[:,0],'CTR_2':ctrvar[:,1]})
contri_variables
```

Out[128]:

	id	CTR_1	CTR_2
0	diagonal	0.005369	0.607837
1	height_left	0.226102	0.115091
2	height_right	0.241888	0.055480
3	margin_low	0.185784	0.102744
4	margin_up	0.124284	0.019915
5	length	0.216572	0.098933

Constatation:

- Les 5 variables sauf diagonal contribuent proportionnellement pas mal à F1
- Diagonal contribue 60% à F2

In [233]:
```
#verification contributions des variables aux axes
print(np.sum(ctrvar,axis=0))

print(np.sum(contri_variables,axis=0))
```

```
[1. 1. 1. 1. 1. 1.]
id      diagonalheight_leftheight_rightmargin_lowmargi...
CTR_1                                                1.0
CTR_2                                                1.0
```

Pour chaque axe, la **somme des contributions de tous ces 6 variables** sur chaque axe est égale à **1**. Ainsi on a [1.1.1.1.1.1] vu qu'on a 6 axes-facteurs.

M2 - Classification des individus: K means

M2 - algorithme de classification K means

Pourquoi K-means ?

- On sait déjà **le nombre de clusters** à sortir est 2
- K-means est **bien liée avec ACP**
- **Efficacité** de méthode K means

* Mais on pourrait aussi employer classification hiérarchique vu le volume d'observations n'est pas grand.

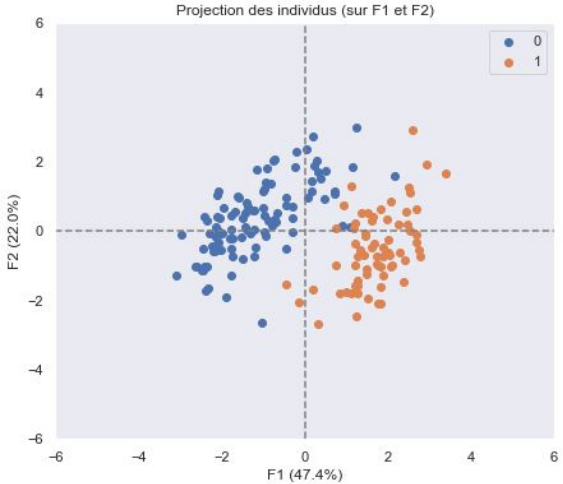
In [301]:
```
notesO.head()
```

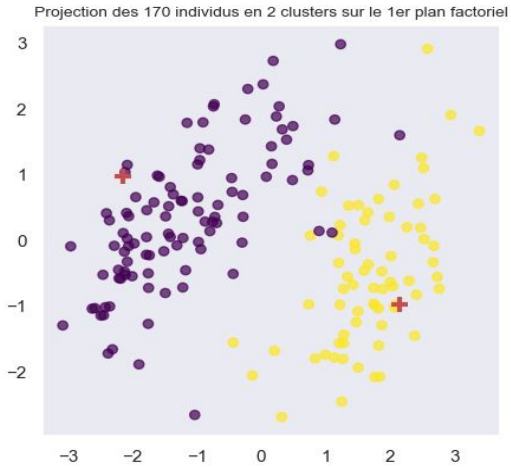
Out[301]:

is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
True	171.81	104.86	104.95	4.52	2.89	112.83
True	171.67	103.74	103.70	4.01	2.87	113.29
True	171.83	103.76	103.76	4.40	2.88	113.84
True	171.80	103.78	103.65	3.73	3.12	113.63
True	172.05	103.70	103.75	5.04	2.27	113.55

In [306]:
```
# Nombre de clusters souhaités
n_clust = 2

# préparation des données pour le clustering
X = notesO.values

# Réduire n'est ici pas nécessaire car les variables sont exprimées dans la même unité
#X_scaled = preprocessing.StandardScaler().fit_transform(X)

# Centrage et Réduction
std_scale = preprocessing.StandardScaler().fit(X)     # centrage
X_scaled = std_scale.transform(X)                     # reduction


# Clustering par K-means
km = KMeans(n_clusters=n_clust)
km.fit(X)

# Récupération des clusters attribués à chaque individu
clusters = km.labels_

# Affichage du clustering par projection des individus sur le premier plan factoriel
pca = decomposition.PCA(n_components=n_comp).fit(X_scaled)
X_projected = pca.transform(X_scaled)
```

M2 - K means
Projection des individus en 2 clusters





Constatation:

Sur le 1er plan factoriel, les 2 clusters-partitions se distinguent clairement.

Vu que le 1er plan représente environ **70%** d'inertie et la plupart des variables (diagonal, length, height_left, height_left, margin_low) y projetent de bonne qualité (COS2), on dirait que **cette projection des clusters sur 1er plan a du sens** et on pourrait y faire confiance.

M2 - K means
Matrice de confusion

Matrice de confusion

col_0	False	True
cluster1	68	1
cluster2	2	99

- 2 faux billets sont classés dans cluster 2 qui représente globalement billets vrais (99/101)
- 1 vrai billet est classé dans cluster 1 qui représente globalement billets faux (68/69)

On sait qu'il existe au total 170 billets dont 100 vrais billets et 70 faux,

donc le taux d'erreur = 3/170 = 1.76%

=> cette K-means clustering qu'on a appliqué est globalement bonne pour ce cas

M3 - Modélisation par régression logistique

M3 - Régression logistique

Est-il correct de choisir le modèle reg logistique?

In [313]:
```
model = LogisticRegression()
model.fit(X,y)                    # entrainer le model avec méthode fit

probabilities= model.predict_proba(X)
probabilities
```

In [317]:
```
notesO['predict'] = model.predict(X)
notesO.head()
```

Out[317]:

is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length	predict
True	171.81	104.86	104.95	4.52	2.89	112.83	True
True	171.67	103.74	103.70	4.01	2.87	113.29	True
True	171.83	103.76	103.76	4.40	2.88	113.84	True
True	171.80	103.78	103.65	3.73	3.12	113.63	True
True	172.05	103.70	103.75	5.04	2.27	113.55	True

In [147]:
```
pd.crosstab(notesO['is_genuine'], notesO['predict'])
```

Out[147]:

is_genuine	False	True
False	69	1
True	1	99

In [327]:
```
print("2/170 individus sont mal placés dans cette matrice, donc le taux d'erreur est 2/170 = ",
      round(2/170, 3))
```

2/170 individus sont mal placés dans cette matrice, donc le taux d'erreur est 2/170 = 0.012

In [328]:
```
from sklearn.metrics import accuracy_score
accuracy_score(notesO['is_genuine'], notesO['predict'])
```

Out[328]: 0.9882352941176471

performance du mode > 98%, donc on pourrait dire que c'est un bon modèle pour la prédiction

M3 - Régression logistique
Entraînement et test de modèle logistique

Split de jeu de données

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,  test_size=0.2, random_state= 42)

print("Training features/target:", X_train.shape, y_train.shape)
print("Testing features/target:", X_test.shape, y_test.shape)
```

```
Training features/target: (136, 6) (136,)
Testing features/target: (34, 6) (34,)
```

Entrainement et test

```
logmodel = LogisticRegression(penalty='l2', C=0.1)
logmodel.fit(X_train, y_train)                    # entrainer logmodel avec train data. méthode fit()

y_pred = logmodel.predict(X_test)                 #prédire y avec X_test
y_pred_proba = logmodel.predict_proba(X_test)[:,1]
print(y_pred)
print(y_pred_proba)
```

20% de jeu de données (test_size=0.2) sont pour le **test**, autrement dit **80%** pour l'**entraînement** de modèle

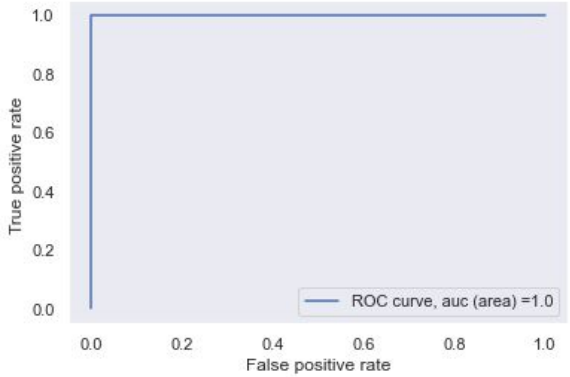
M3 - Régression logistique

Niveau de précision de logmodel

```
print("Accuracy score of logmodel for test data: ",
      round(metrics.accuracy_score(y_test, y_pred), 4))

# ici y_test est dataset existant de y pour test, y_pred est le y prédit par logmodel via x_test
```

Accuracy score of logmodel for test data: 0.9706

> 97%: logmodel est performant



AUC =1, ctd. Il y a 100% de chance que logmodel est en mesure de distinguer les individus de classe positive de ceux de classe négative.

M3 - Régression logistique

Optimisation de choix de variables avec RFE

```
from sklearn.datasets import make_friedman1
from sklearn.feature_selection import RFE

selector = RFE(logmodel, step=1)                    #recursive feature elimination (RFE) is to select features by recursiv
selector = selector.fit(X_train, y_train)
selector.support_
```

```
array([False, False,  True,  True, False,  True])
```

```
print("Selon selector, ces variables sont choisies pour la prise de décision False/True: ",  np.array(notes.columns)[[2,3,-1]])
```

```
Selon selector, ces variables sont choisies pour la prise de décision False/True: ['height_right' 'margin_low' 'length']
```

```
print(selector.predict(X_test))

print("proba False I  proba True: ")
print(selector.predict_proba(X_test))
```

```
[False  True False  True False False  True False  True  True False
 False  True  True  True  True False  True False  True  True  True
  True  True  True  True  True  True  True  True False False]
proba False I  proba True:
[[0.71481126 0.28518874]
 [0.09752728 0.90247272]
 [0.77331092 0.22668908]
 [0.06721685 0.93278315]
 [0.57902677 0.42097323]
```

```
print("Matrice de confusion_selector")
pd.crosstab(selector.predict(X_test),y_test)
```

```
Matrice de confusion_selector
```

is_genuine	False	True
row_0		
False	12	0
True	0	22

```
print("Accuracy score of selector for test data: ", metrics.accuracy_score(selector.predict(X_test),
# selector: logmodel avec 'features' optimisés
```

```
Accuracy score of selector for test data: 1.0
```

Algorithme **selector** arrive à distinguer les vrais et faux billets à **100%**

M4 - Cas pratique: test de l’algorithme

M4 - Cas pratique: simulation - test de l'algorithme

n [403]:
```
example = pd.read_csv("example.csv")
example.head()
```

ut[403]:

	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

n [404]:
```
from sklearn.linear_model import LogisticRegression
XX = example.drop(['id'], axis=1).values
XX
```

ut[404]:
```
array([[171.76, 104.01, 103.54,   5.21,   3.3 , 111.42],
       [171.87, 104.17, 104.13,   6. ,   3.31, 112.09],
       [172. , 104.58, 104.29,   4.99,   3.39, 111.57],
       [172.49, 104.55, 104.34,   4.44,   3.03, 113.2 ],
       [171.65, 103.63, 103.56,   3.77,   3.16, 113.33]])
```

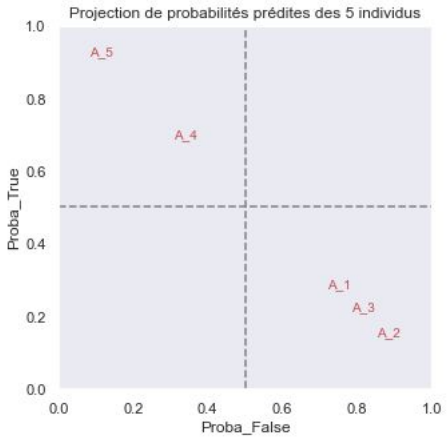
n [405]:
```
probability =  selector.predict(XX)
proba_percentage =  selector.predict_proba(XX)
print(probability)
print("False % , True %:")
print(proba_percentage)
```

```
[False False False  True  True]
False % , True %:
[[0.71976459 0.28023541]
 [0.85286893 0.14713107]
 [0.78312751 0.21687249]
 [0.30729561 0.69270439]
 [0.08078442 0.91921558]]
```

:
```
df = pd.DataFrame({'ID': example['id'], "Genuine": probability,
                   "Proba_False": proba_percentage[:,0],
                   "Proba_True": proba_percentage[:,1]})
df
```

:

	ID	Genuine	Proba_False	Proba_True
0	A_1	False	0.719765	0.280235
1	A_2	False	0.852869	0.147131
2	A_3	False	0.783128	0.216872
3	A_4	True	0.307296	0.692704
4	A_5	True	0.080784	0.919216



Q & A

MERCI !

Xuefei ZHANG, parcours Data Analyst, Jan 2022