P6 : DÉTECTER DES FAUX BILLETS

MOHAMMED MOKEDDEM

MINISTÈRE
DE L'INTÉRIEUR

Liberté
Égalité
Fraternité

OFFICE CENTRAL POUR LA RÉPRESSION DU FAUX MONNAYAGE
DCPJ

SOMMAIRE

1. Description des données (analyses univariées et bivariées)
2. Analyse en Composantes Principales : détail de l'analyse
3. Analyse de la classification par KMeans
4. Détails de la modélisation par régression logistique
5. Conclusion

1. DESCRIPTION DES DONNÉES(ANALYSES UNIVARIÉES ET BIVARIÉES)

Entrée [1954]:

```
1 import pandas as pd
2 import numpy as np
3 notes=pd.read_csv("notes.csv", sep=",")
4 notes
```

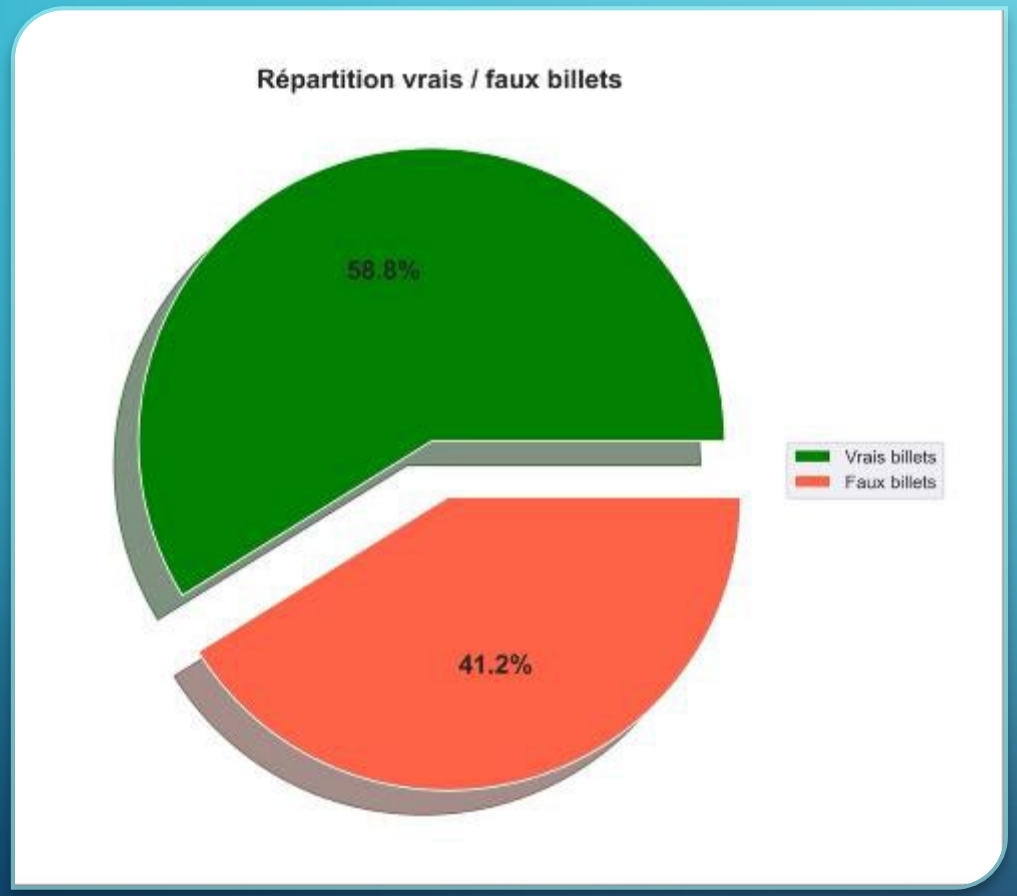
Out[1954]:

	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
...	...	...	...	...	...	...	...
165	False	172.11	104.23	104.45	5.24	3.58	111.78
166	False	173.01	104.59	104.31	5.04	3.05	110.91
167	False	172.47	104.27	104.10	4.88	3.33	110.68
168	False	171.82	103.97	103.88	4.73	3.55	111.87
169	False	171.96	104.00	103.95	5.63	3.26	110.96

170 rows × 7 columns

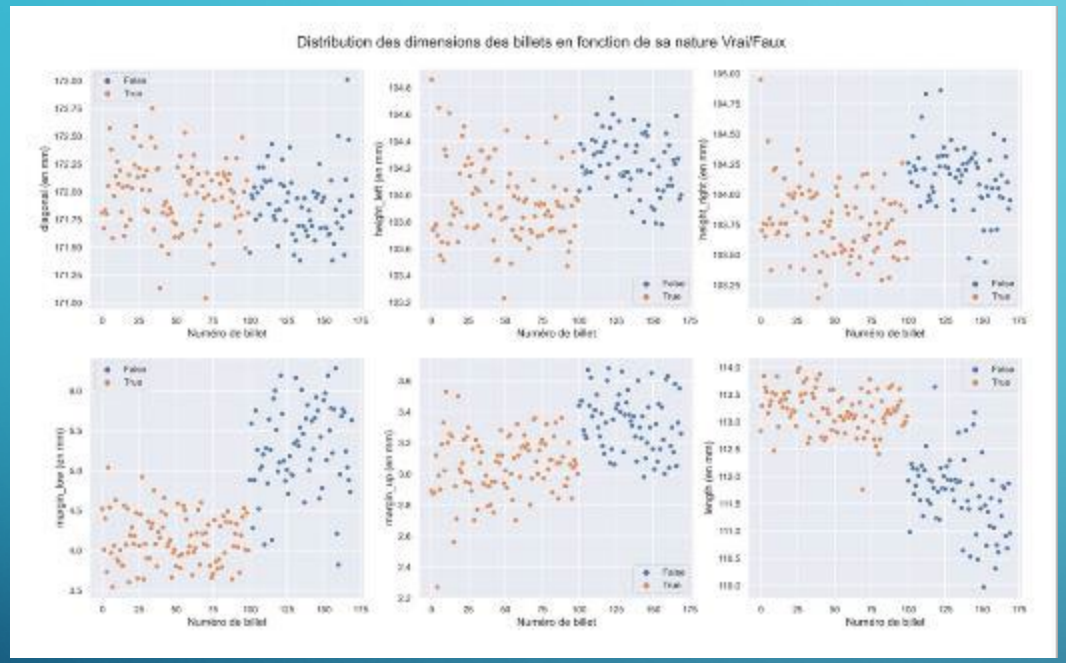
- Jeu de données composé de 6 variables quantitatives et une variable catégorielle
- Il y a 170 individus (billets de banque)
- La variable catégorielle de type booléen “is_genuine” prend la valeur “True” ou “False” selon que le billet soit vrai ou faux
- Les variables quantitatives représentent les dimensions de chaque billet en mm, ce sont des reels.

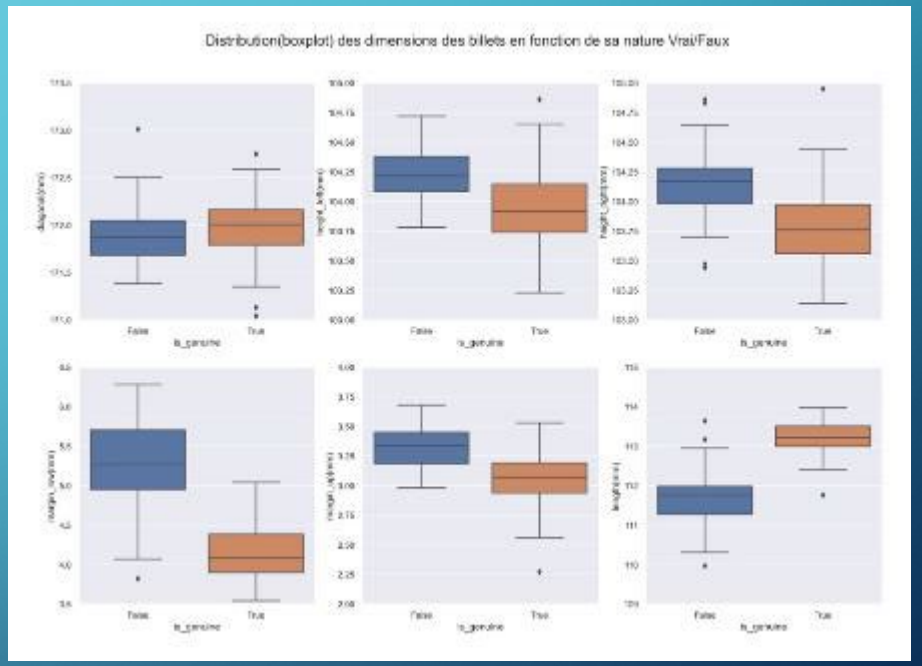
1. DESCRIPTION DES DONNÉES(ANALYSES UNIVARIÉES ET BIVARIÉES)



- Il y a 100 vrais billets et 70 faux billets.
- Repartition proche de 60/40 %
- Répartition intéressante pour la suite de l'analyse (Kmeans)
- Pas de valeurs aberrantes, manquantes ni de données dupliquées

1. DESCRIPTION DES DONNÉES(ANALYSES UNIVARIÉES ET BIVARIÉES)





- Certaines variables présentent des distributions différentes selon que le billet soit « vrai » ou « faux » :
  - Moyenne ou médiane ≠
  - Ecart-type ≠

1. DESCRIPTION DES DONNÉES(ANALYSES UNIVARIÉES ET BIVARIÉES)

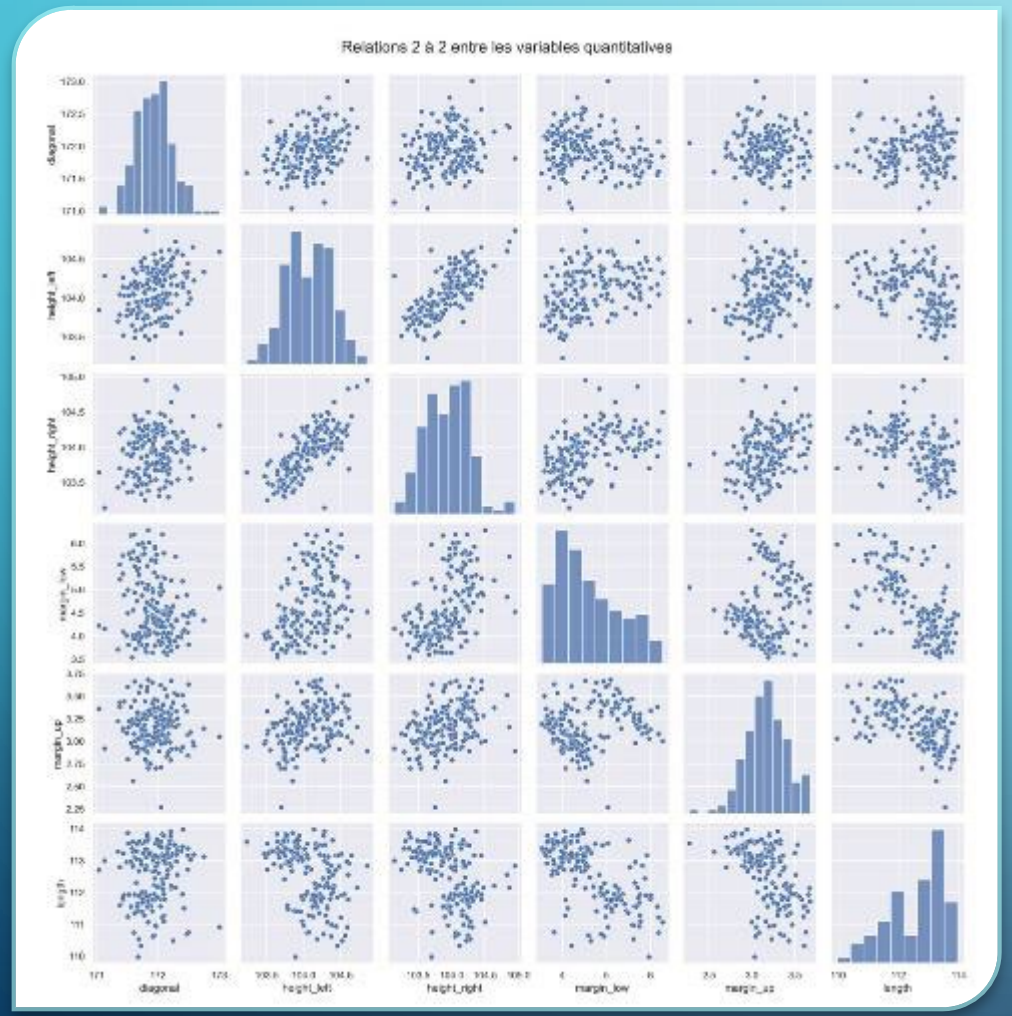
- Comparaison des moyennes :
  - Test de Student
  - Hypothèse H0 : Toutes les populations ont la même moyenne
- Si les populations "vrai" et "faux" ont la meme moyenne alors la variable n'est pas pertinente pour la détection de faux billets.

	diagonal	height_left	height_right	margin_low	margin_up	length
Student_Statistic	1.82229	-6.74697	-8.56499	-17.2886	-9.29587	18.9751
Student_pvalue	0.0701897	2.3342e-10	6.66525e-15	3.94015e-39	7.56739e-17	1.23482e-43
Student_Hypothèse H0	Non rejetée	Rejetée	Rejetée	Rejetée	Rejetée	Rejetée

L'ensemble des hypothèses H0 est rejetée excepté pour la variable "diagonal".

La variable "diagonal" ne permet pas de détecter de manière exacte les faux billets.

1. DESCRIPTION DES DONNÉES(ANALYSES UNIVARIÉES ET BIVARIÉES)

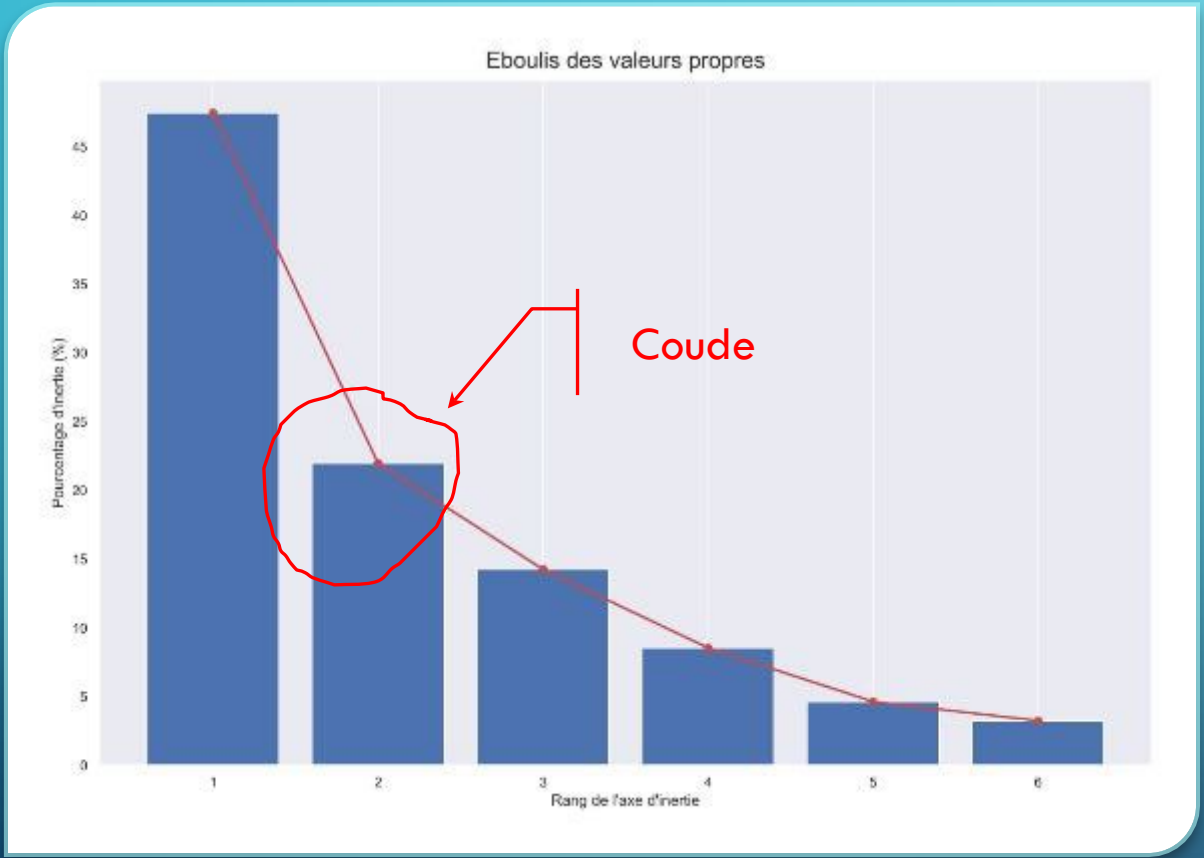


- On remarque une relation qui semble linéaire entre "height_left" et "height_left"
- Graphiquement, il n'est pas clair qu'il y aient d'autres relations linéaires.
- Vérifions la force de la relation en affichant la matrice des corrélations de Pearson :

	diagonal	height_left	height_right	margin_low	margin_up	length
diagonal	1.0	0.32	0.22	-0.18	-0.027	0.08
height_left	0.32	1.0	0.73	0.42	0.32	-0.42
height_right	0.22	0.73	1.0	0.51	0.37	-0.42
margin_low	-0.18	0.42	0.51	1.0	0.17	-0.64
margin_up	-0.027	0.32	0.37	0.17	1.0	-0.53
length	0.08	-0.42	-0.42	-0.64	-0.53	1.0

- Il existe une relation linéaire forte entre "height_left" et "height_left".

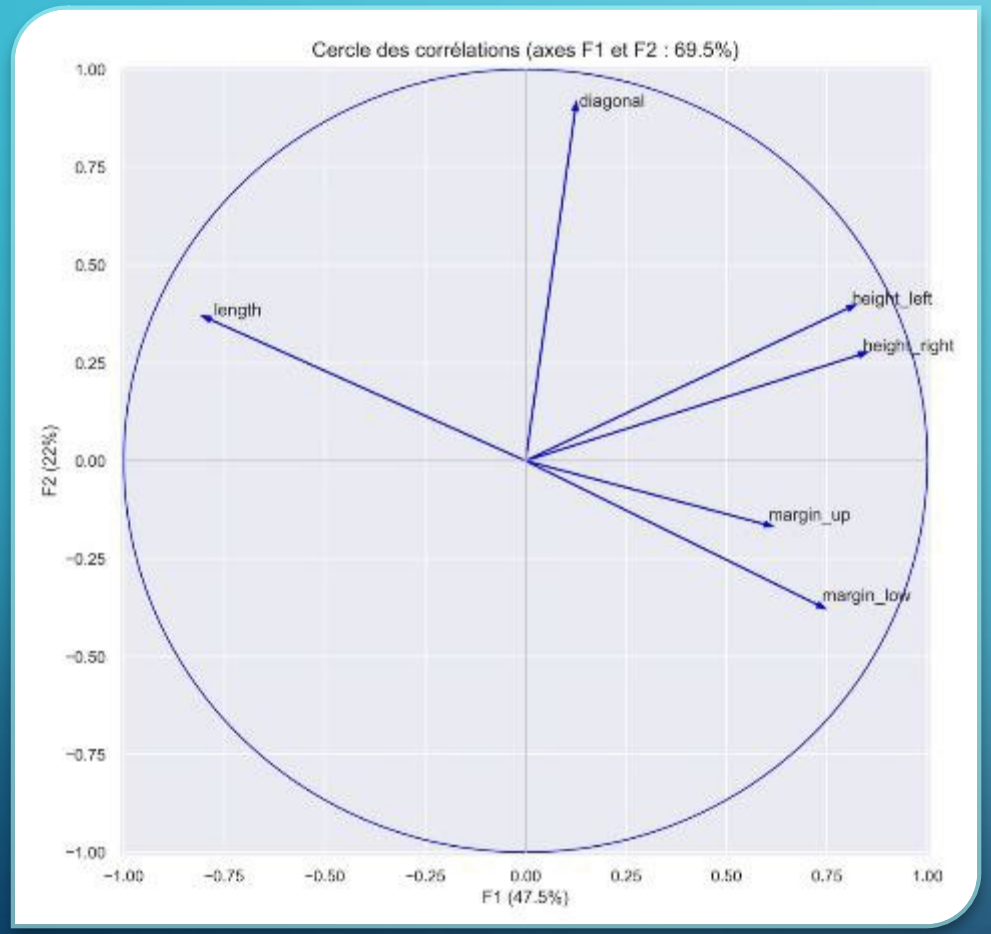
2. ACP : EBOULIS DES VALEURS PROPRES



- Pour l'ACP, je selectionne toutes mes variables quantitatives (6)
- Centrer et réduire les données car ordres de grandeurs des ≠ variables trop disparates
- Les composantes 1 et 2 expliquent 69.4% de la dispersion du nuage d'individus => satisfaisant au regard de l'objectif de l'ACP.

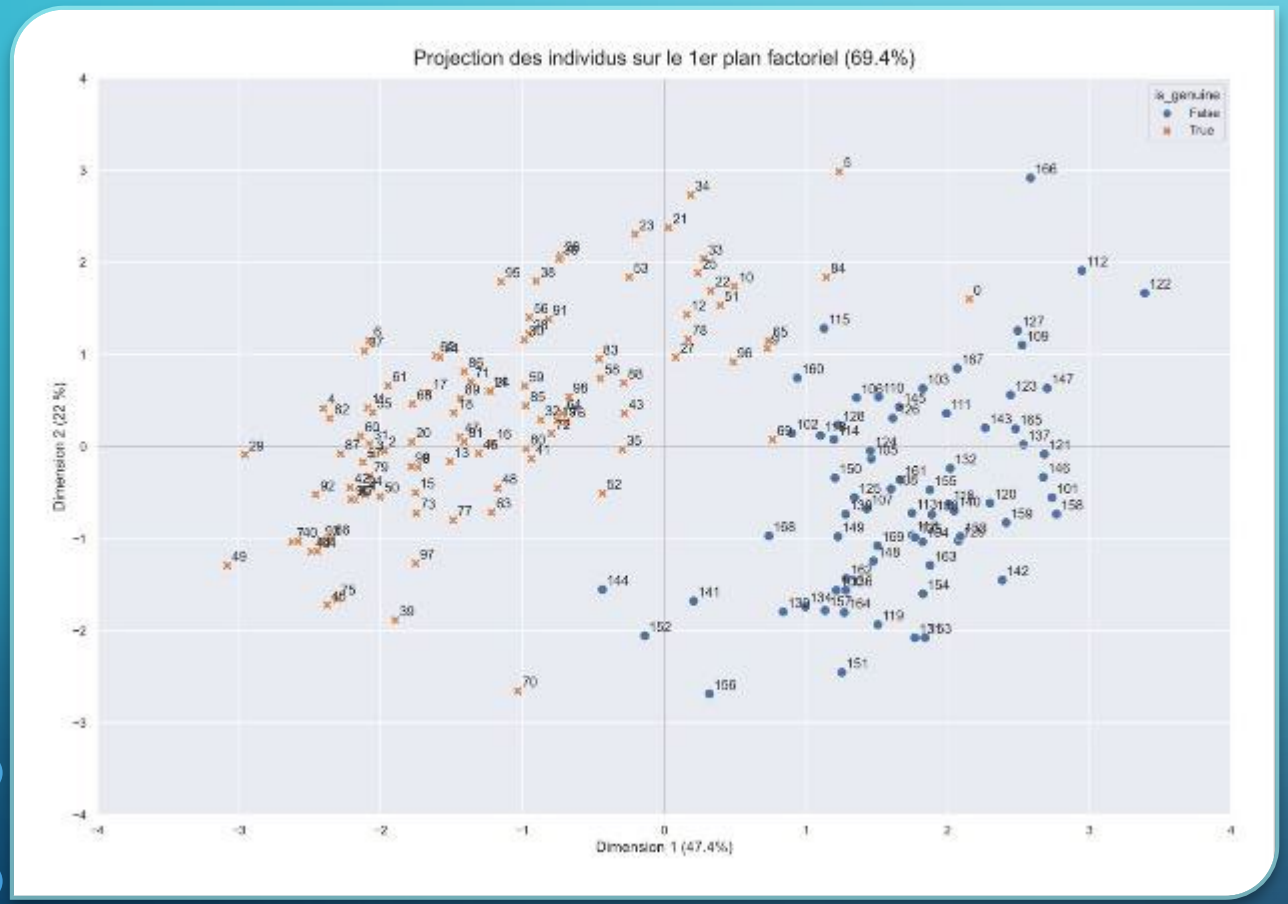
	Dimension	Variance expliquée	% variance expliquée	% cum. var. expliquée
0	Dim1	2.846875	47.4	47.4
1	Dim2	1.317426	22.0	69.4
2	Dim3	0.854071	14.2	83.6
3	Dim4	0.511578	8.5	92.2
4	Dim5	0.276769	4.6	96.8
5	Dim6	0.193280	3.2	100.0

2. ACP : CERCLE DES CORRÉLATIONS



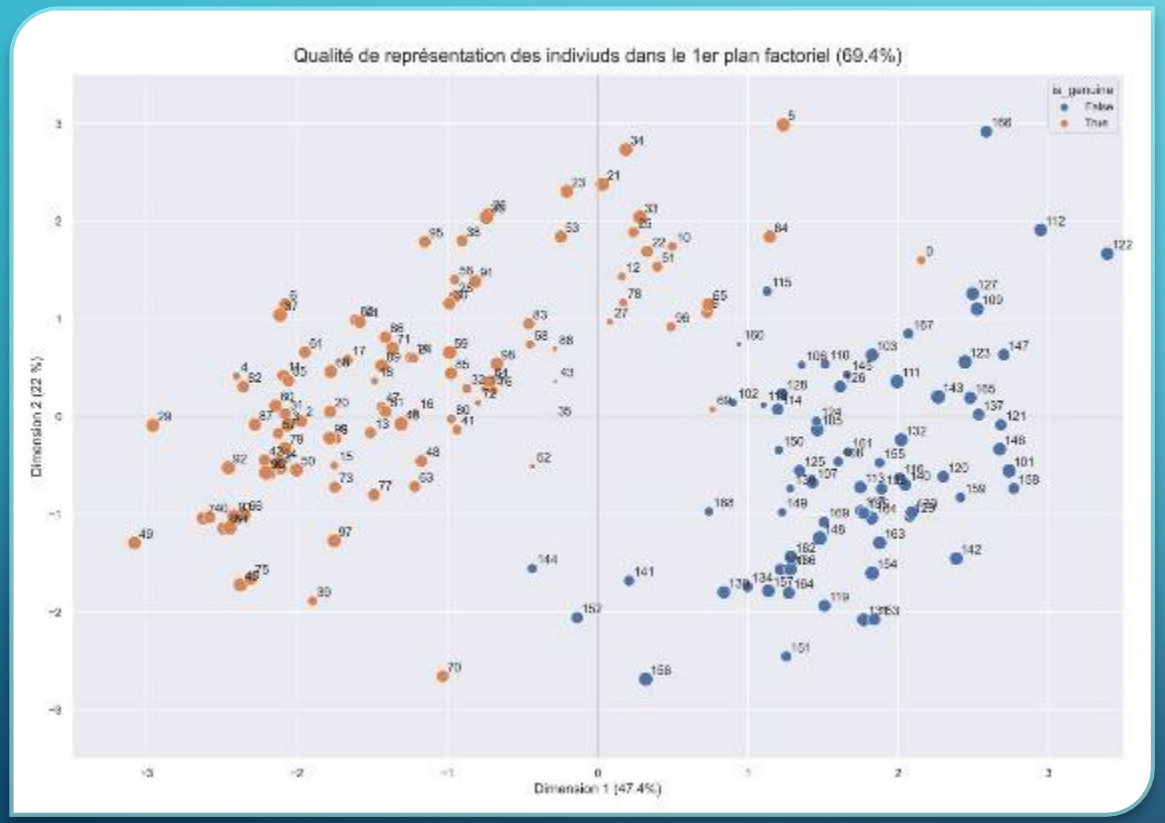
- "height_left","height_right", "margin_low" et "length" sont bien corrélées à F1
- "diagonal" est très bien corrélée à F2
- "margin_up" n'est pas bien représentée sur le plan malgré une très bonne corrélation à F1(faible angle)
- "height_left" et "height_right" sont très bien corrélées entre elles
- La variable F1 synthétisent 5 variables initiales : elle traduit la nature du billet

2. ACP : PROJECTION DES INDIVIDUS SUR LE 1ER PLAN FACTORIEL



- La projection sur le premier plan factoriel met bien en lumière les deux clusters "Vrais " et "Faux "
- On voit que les vrais billets sont plutôt à gauche de l'axe F1 et les faux billets à droite de l'axe F1

2. ACP : QUALITÉ DE REPRÉSENTATION DES INDIVIDUS



- La qualité de représentation des individus = la qualité de projection sur le premier plan
- Si 2 individus sont bien projetés, alors leur distance en projection est proche de leur distance dans l'espace.
- Deux points proches l'un de l'autre sur le graphique peuvent correspondre à des individus éloignés l'un de l'autre
- EXEMPLE : 69 et 102 sont des billets éloignés l'un de l'autre (l'un est vrai et l'autre faux) mais proches sur le plan .
- Ils ont un COS2 total faible (billet 69=0.241648 et billet 102=0.335342) ce qui signifie qu'ils sont mal projetés

2. ACP : CONTRIBUTION DES INDIVIDUS AUX AXES

Entrée [1851]:

```
# J'identifie le nombre d'individus ayant une contribution importante

CTR_importante= CTR[(CTR["CTR_1"]>0.023528) | (CTR["CTR_2"]>0.023528) ]
CTR_importante
```

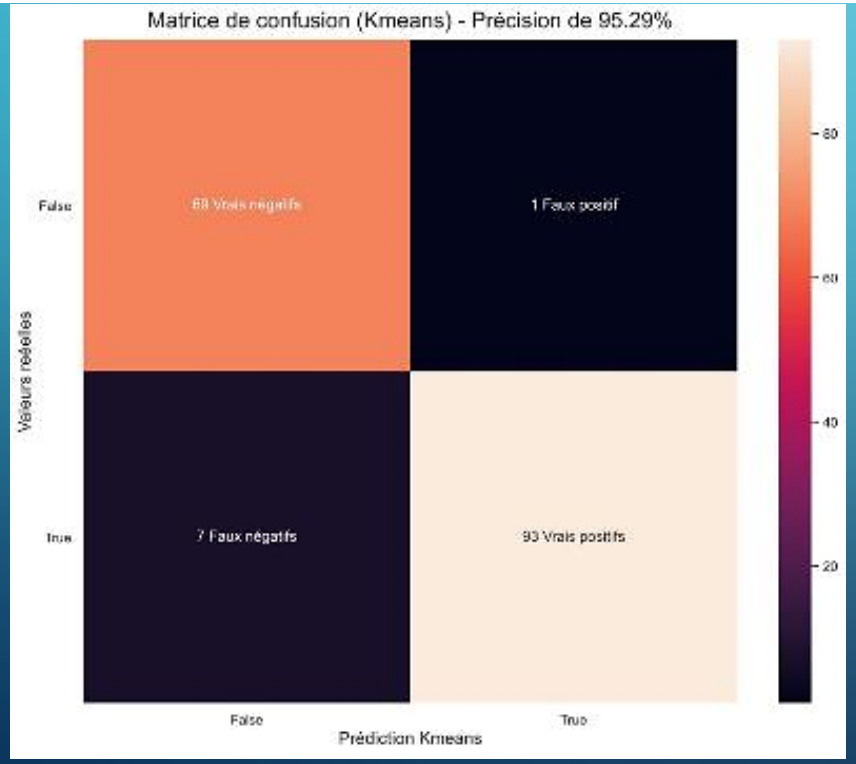
Out[1851]:

	id	CTR_1	CTR_2	is_genuine
5	5	0.003155	0.039736	True
21	21	0.000002	0.025151	True
23	23	0.000088	0.023682	True
34	34	0.000073	0.033302	True
70	70	0.002197	0.031516	True
122	122	0.023758	0.012372	False
151	151	0.003251	0.026922	False
156	156	0.000211	0.032264	False
166	166	0.013811	0.037927	False

- Quels sont les individus qui participent le plus à la formation de l'axe ?
- Elle est calculée pour chaque axe
- Contribution importante si : CTR>4xCTRmoy= 4x0.005882=0.023528
- Individu sur-représenté si CTR > 0.25 ==> Retirer l'individu
- Aucun individu n'est sur-représenté
- 1 individu à une CTR importante à F1
- 8 individus ont une CTR importante à F2
- Je garde tous les individus (ils sont bien représentés)

3. KMEANS : MATRICE DE CONFUSION

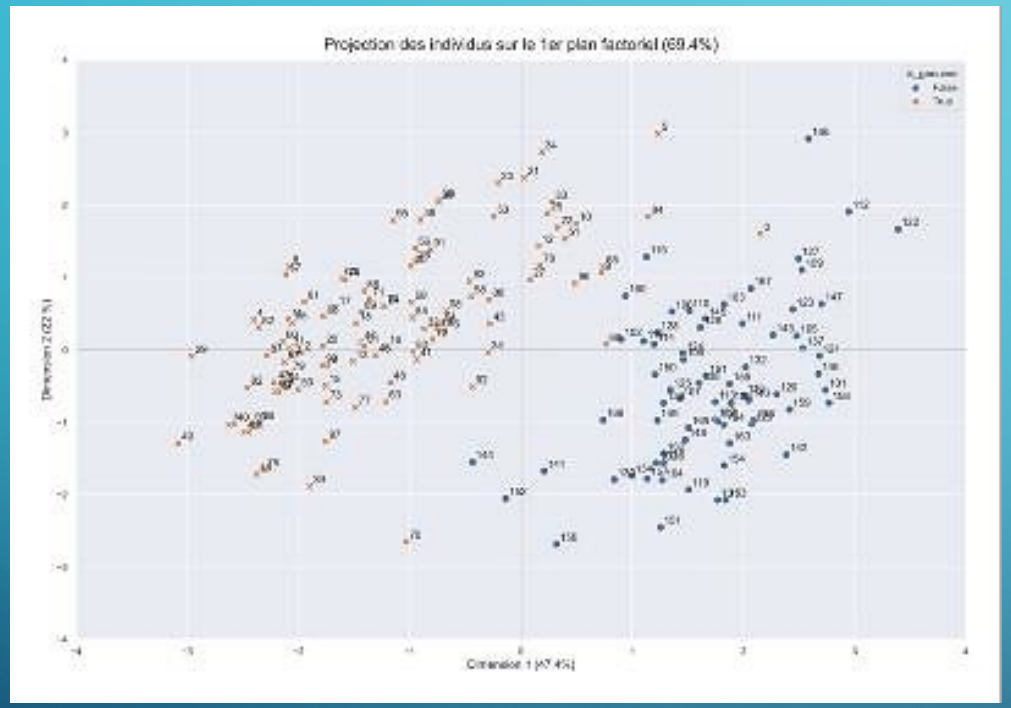
```
Entrée [1957]:
1 # Récupération des clusters attribués à chaque individu
2 clusters = km.labels_
3 clusters

Out[1957]: array([0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 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, 0,
                  1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                  1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```



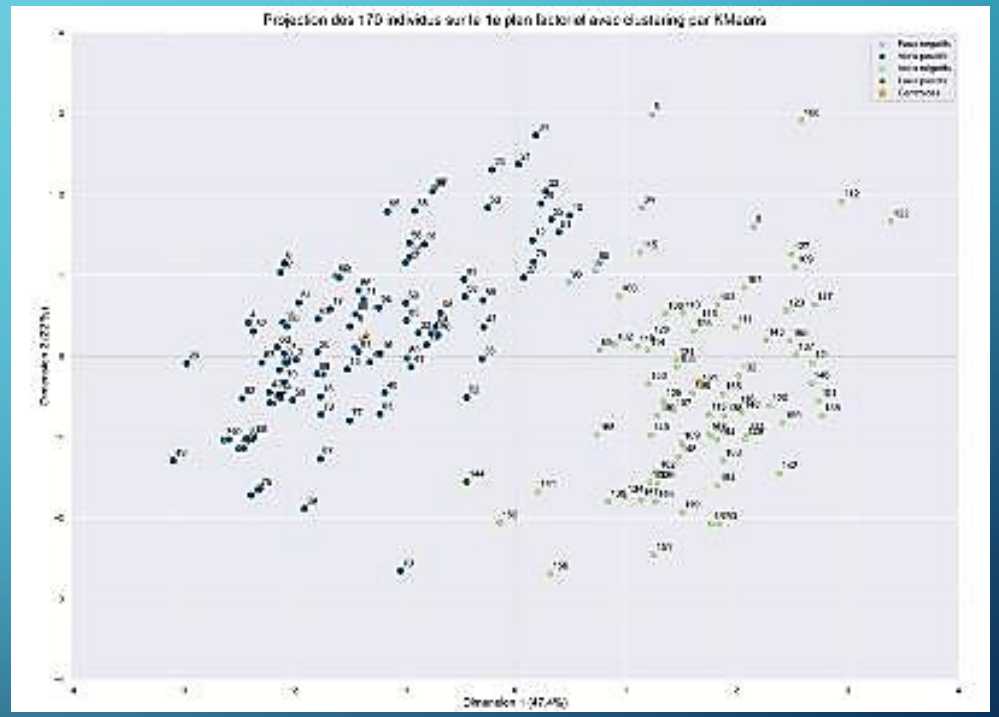
- Le nombre de clusters est déterminé par le découpage naturel de notre jeu de données (Vrai/Faux)
- On obtient ainsi un partitionnement des données en 2 clusters
- On compare les données du Kmeans avec la variable is_genuine
- On obtient une précision de 95.29% avec ce clustering

3. KMEANS : PROJECTION DES INDIVIDUS DANS LE 1ER PLAN FACTORIEL

ACP

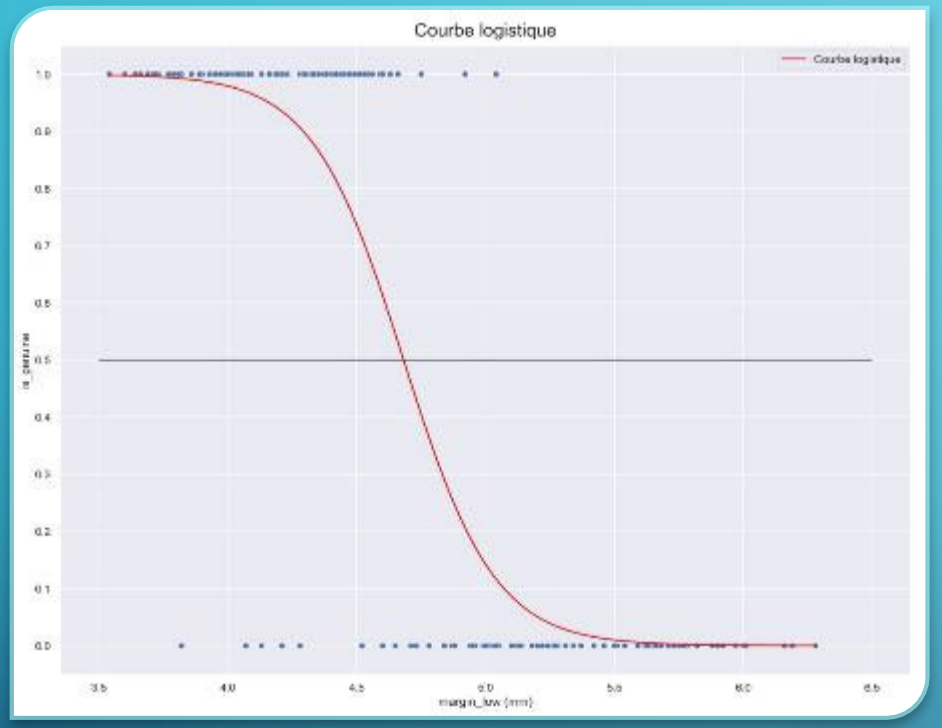


KMEANS



- La répartition des clusters calculés par l'algorithme Kmeans est fidèle à la projection de l'ACP
- Les « Faux négatifs » et « Faux positifs », identifiés, sont sur la frontière Vrai/Faux de la projection

4. RÉGRESSION LOGISTIQUE : REPRÉSENTATION GRAPHIQUE



Generalized Linear Model Regression Results

Dep. Variable:	is_genuine	No. Observations:	170
Model:	GLM	Df Residuals:	168
Model Family:	Binomial	Df Model:	1
Link Function:	logit	Scale:	1.0000
Method:	IRLS	Log-Likelihood:	-40.056
Date:	Wed, 22 Sep 2021	Deviance:	80.112
Time:	12:13:45	Pearson chi2:	245.
No. Iterations:	7		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	26.4357	4.136	6.391	0.000	18.329	34.542
margin_low	-5.6461	0.898	-6.287	0.000	-7.406	-3.886

On obtient les paramètres estimés : $\hat{\beta}_1 = 26.4357$ et $\hat{\beta}_2 = -5.6461$.

Je les enregistre :

Entrée [1972]:
```
1 beta1 = reg_log1.params[0]
2 beta2 = reg_log1.params[1]
```

Dans le but de tracer la courbe logistique entre les abscisses $x = 3.54$ et $x = 6.28$, on définit une séquence de 3.54 à 6.28 par pas de 170, puis on la place dans la variable x. On calcule ensuite les ordonnées de la courbe, grâce à l'expression de la courbe sigmoïde en S :

$$P(x) = \frac{e^{\beta_1 + \beta_2 x}}{1 + e^{\beta_1 + \beta_2 x}}$$

- Modèle qui permet de prédire la probabilité qu'un évènement arrive ou pas à partir de l'optimisation des coefficients de regression
- Statmodel nous permet d'obtenir les résultats de la regression logistique sur la variable "margin_low"[15]de façon à construire la courbe logistique qui nous permettra de prédire l'évènement

4. RÉGRESSION LOGISTIQUE : CRÉATION DU MODÈLE

Entrée [1978]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

y=notes['is_genuine']
X=notes.loc[:,["diagonal","height_left","height_right",'margin_low','margin_up','length']]

#fractionner dataset (train-test)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2, random_state = 0)
print('Train:',X_train.shape, 'Test:',X_test.shape)

#Centrage et réduction
scaler=StandardScaler()# crée un scaler
scaler.fit(X_train) # calcule mu et sigma sur X_train uniquement
X_train_scaled=scaler.transform(X_train)
X_train_scaled=pd.DataFrame(X_train_scaled,index=X_train.index, columns=X_train.columns)

X_test_scaled=scaler.transform(X_test)
X_test_scaled=pd.DataFrame(X_test_scaled,index=X_test.index, columns=X_test.columns)

#instanciation du modèle
modele_regLog = linear_model.LogisticRegression()
#training
modele_regLog.fit(X_train_scaled, y_train)


#calcule de précision du jeu d'entrainement

print("La précision du modèle sur le jeu d'entrainement:{:.3f}".format(modele_regLog.score(X_train_scaled, y_

#calcule de précision du jeu de test
print("La précision du modèle sur le jeu de test:{:.3f}".format(modele_regLog.score(X_test_scaled, y_test)))
```

Entrée [1985]:

```
#Calcul de  la précision de mon modèle sur le  jeu de données initiales "notes":
X_notes_scaled=scaler.transform(X)
print("La précision du modèle sur le jeu de données initiales 'notes' :{:.3f}".format(modele_regLog.score(X_
```

```
La précision du modèle sur le jeu de données initiales 'notes' :0.994
```

- Regression logistique sur l'ensemble des variables du dataset
- Fractionnement du dataset en deux jeux : entrainement et test (80/20 %)
- Le jeu d'entrainement sert à entrainer mon modèle, le jeu de test à verifier la precision de mon modèle
- Les données sont centrées réduites
- Précision :
  - Train : 99.3%
  - Test : 100%
- Modèle excellent
- Test du modèle sur le jeu de données initiales ("notes.csv") :
  - Résultats : 99.4 %
- Meilleure prédiction que Kmeans

4. RÉGRESSION LOGISTIQUE : COEFFICIENTS BETA DE MON MODÈLE

Nous avons vu, dans le cas d'une régression logistique avec une seule variable que :

$$P(x) = \frac{e^{\beta_1+\beta_2 x}}{1+e^{\beta_1+\beta_2 x}}$$

Dans une situation de variables explicatives multiples l'équation se généralise en :

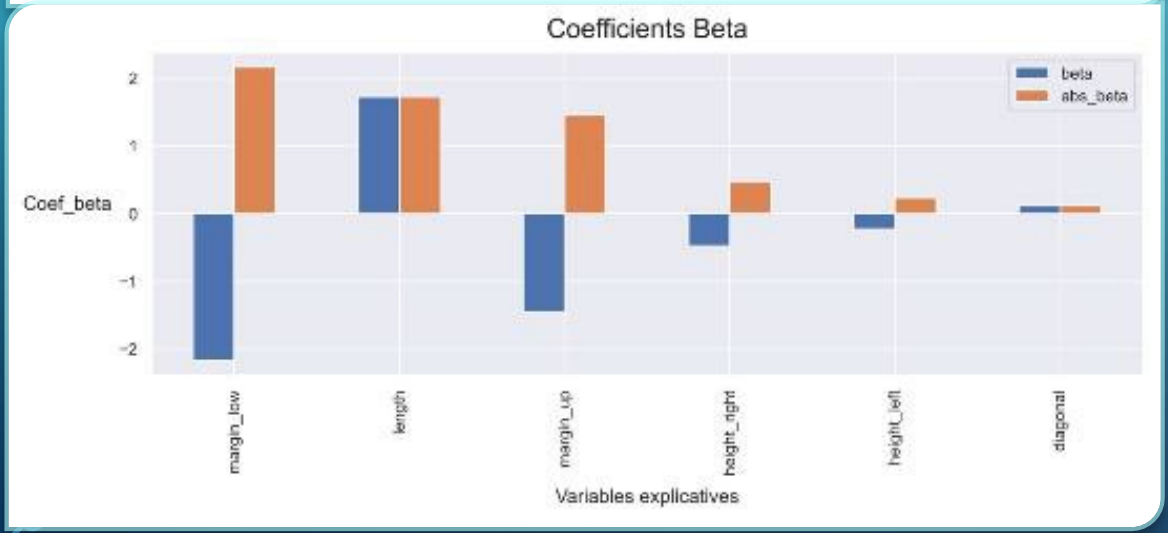
$$P(x) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_k X_k}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_k X_k}} = \frac{exp(\sum \beta X)}{1+exp(\sum \beta X)}$$

Le modèle précédent n'est pas linéaire dans l'expression des paramètres beta_X puisque la probabilité de réalisation ne s'exprime pas comme une addition des effets des différentes variables explicatives.

Pour obtenir un tel modèle (linéaire dans ses paramètres), il est nécessaire de passer par une transformation logit :

$$logit(p) = log(\frac{p}{1-p}) = \sum_{j=1}^{n} \beta_j X_j$$

Cette transformation logit est la fonction de lien qui permet de mettre en relation la probabilité de réalisation (bornée entre 0 et 1), et la combinaison linéaire de variable explicatives.



- Regression logistique sur 1 seule variable => 2 coef beta
- Pour une regression logistique sur plusieurs variables, l'équation générale doit passer par une transformation logit (fonction de lien)
- Les parametres Beta de chaque variable explicative sont représentés graphiquement
- Plus la valeur absolue de beta est grande plus la variable influence le modèle
- Les variables « margin_low », « length » et « margin-up » influencent fortement le modèle
- Prédiction sur ces 3 variables

4. RÉGRESSION LOGISTIQUE : CRÉATION DU MODÈLE OPTIMISÉ

Entrée [1993]:

```
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler

y=notes['is_genuine']
X=notes.loc[:,['margin_low','margin_up','length']]

#fractionner dataset (train-test)
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2, random_state = 42)
print('Train:',X_train.shape, 'Test:',X_test.shape)

#Centrage et réduction
scaler=StandardScaler()# crée un scaler
scaler.fit(X_train) # calcule mu et sigma sur X_train uniquement
X_train_scaled=scaler.transform(X_train)
X_train_scaled=pd.DataFrame(X_train_scaled,index=X_train.index, columns=X_train.columns)

X_test_scaled=scaler.transform(X_test)
X_test_scaled=pd.DataFrame(X_test_scaled,index=X_test.index, columns=X_test.columns)

#instanciation du modèle
modele_regLog = LogisticRegression()
#training
modele_regLog.fit(X_train_scaled, y_train)


#calcule de précision du jeu d'entrainement

print("La précision du modèle sur le jeu d'entrainement:{:.3f}".format(modele_regLog.score(X_train_scaled, y_

#calcule de précision du jeu de test
print("La précision du modèle sur le jeu de test:{:.3f}".format(modele_regLog.score(X_test_scaled, y_test)))

```

```
Train: (136, 3) Test: (34, 3)
La précision du modèle sur le jeu d'entrainement:0.993
La précision du modèle sur le jeu de test:1.000
```

Entrée [2005]:

```
#Calcul de  la précision de mon modèle sur le  jeu de données initiales "notes":
X_notes_scaled=scaler.transform(X)
print("La précision du modèle sur le jeu de données initiales 'notes' :{:.3f}".format(modele_regLog.score(X_

```

```
La précision du modèle sur le jeu de données initiales 'notes' :0.994
```

- Je modifie mon modèle de Regression logistique en utilisant que 3 variables
- Précision :
  - Train : 99.3%
  - Test : 100%
- Modèle avec 3 variables aussi bon qu'avec 6
- Test du modèle sur le jeu de données initiales ("notes.csv") :
  - Résultats : 99.4 %
- J'utiliserai ce modèle pour le test de l'examinateur

5. CONCLUSION

- Le modele de regression logistique est plus précis que la classification non-supervisée
- Pour notre mission de detection de faux, la regression logistique est plus adaptée