Algorithme de détection de faux billets

...

Projet 6 - Nalron Décembre 2019
OpenClassrooms - ENSAE-ENSAI Formation Continue

L’office central pour la répression du faux monnayage

OCRFM créé le 11 septembre 1929. Création et gestion de deux applications informatiques importantes, à vocation opérationnelle :

- Répertoire automatisé pour l'analyse des contrefaçons de l'euro (***RAPACE***), destiné à permettre à tous les enquêteurs d'identifier les billets contrefaits en euro.
- Fichier national du faux monnayage (***FNFM***), recensant l'ensemble des affaires de fausse monnaie commises sur le territoire national et servant de base de données de documentation et d'analyse opérationnelle.

Démarche et méthodologie

Toutes les étapes, et tous les choix seront détaillés durant l'analyse.

- Transmission des données par la PJ : *jeu de données*
- Traitement des données en **langage Python** support Jupyter Notebook.
- Brève description des données (**analyses univariées** et **bivariées**).
- Analyse en composantes principales (**ACP**)de l'échantillon.
- Classification par l'algorithme **K-Means**.
- **Visualisation** de la partition obtenue dans le 1er plan factoriel de l'ACP.
- Modélisation des données à l'aide d'une **régression logistique**.
- **Prédiction** possible sur d'autres jeux de données.

Les données : échantillon de 170 observations

Nous avons **6 variables quantitatives continues** :

- la longueur du billet (en mm),
- la hauteur du billet (mesurée sur le côté gauche, en mm),
- la hauteur du billet (mesurée sur le côté droit, en mm),
- la marge entre le bord supérieur du billet / l'image de celui-ci (en mm),
- la marge entre le bord inférieur du billet / l'image de celui-ci (en mm),
- la diagonale du billet (en mm).

Ainsi qu'une **variable booléenne pour préciser la nature du billet**.

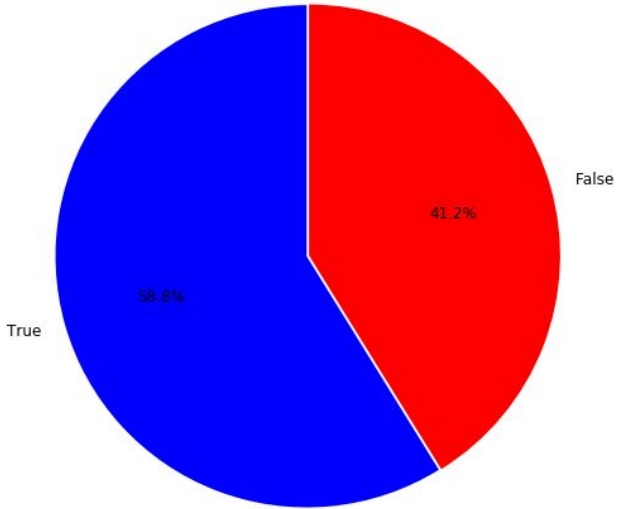
	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	True	171.81	104.86	104.95	4.52	2.89	112.83
1	True	171.67	103.74	103.70	4.01	2.87	113.29
2	True	171.83	103.76	103.76	4.40	2.88	113.84
3	True	171.80	103.78	103.65	3.73	3.12	113.63
4	True	172.05	103.70	103.75	5.04	2.27	113.55

Analyse des variables

Aperçu de la nature des billets de l'échantillon, ainsi que quelques valeurs descriptives des variables quantitatives.

Aucune valeur manquante, aucune valeur négative, aucun doublon et aucun outlier n'a été identifié dans le jeux de données.

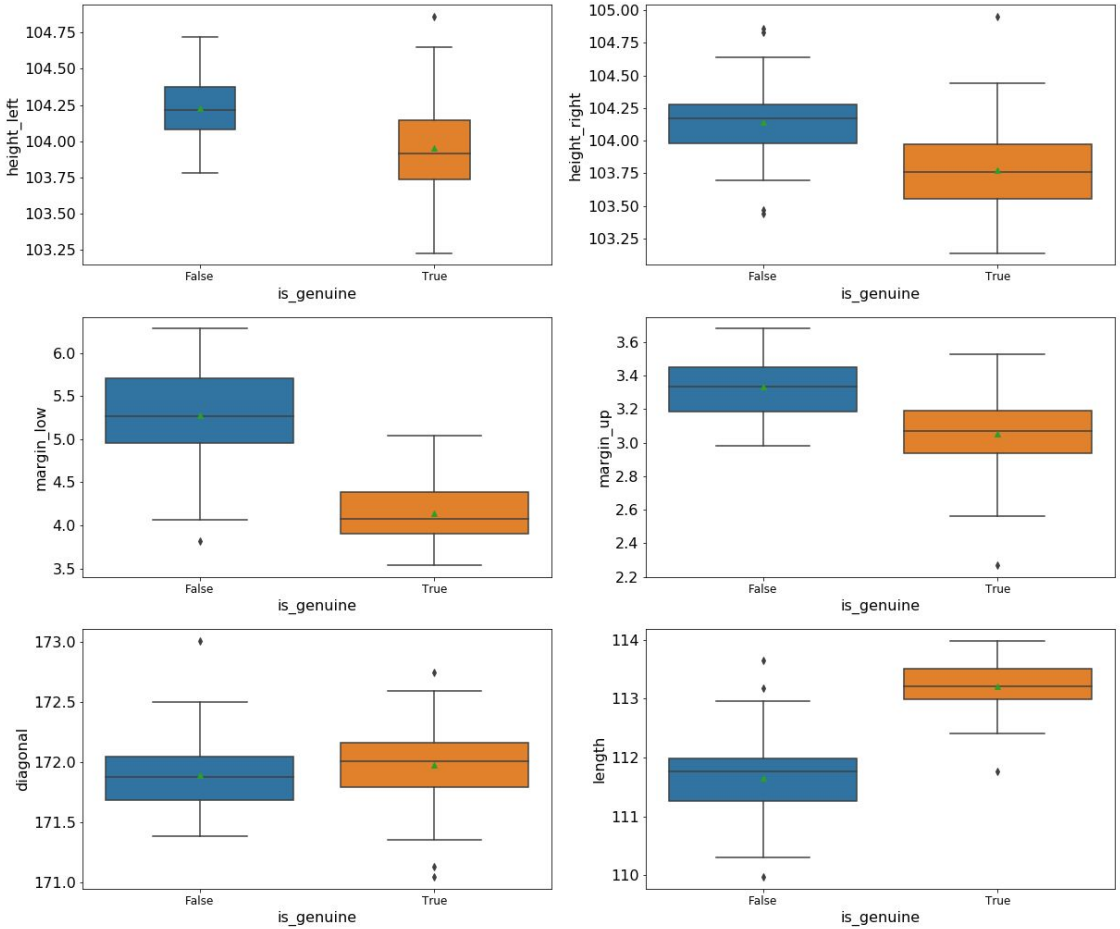
Répartition de la nature Vrai/Faux du billet



	count	mean	std	min	25%	50%	75%	max
diagonal	170.0	171.940588	0.305768	171.04	171.7300	171.945	172.1375	173.01
height_left	170.0	104.066353	0.298185	103.23	103.8425	104.055	104.2875	104.86
height_right	170.0	103.928118	0.330980	103.14	103.6900	103.950	104.1700	104.95
margin_low	170.0	4.612118	0.702103	3.54	4.0500	4.450	5.1275	6.28
margin_up	170.0	3.170412	0.236361	2.27	3.0125	3.170	3.3300	3.68
length	170.0	112.570412	0.924448	109.97	111.8550	112.845	113.2875	113.98

Comparaison de la distribution des variables selon la nature des billets.

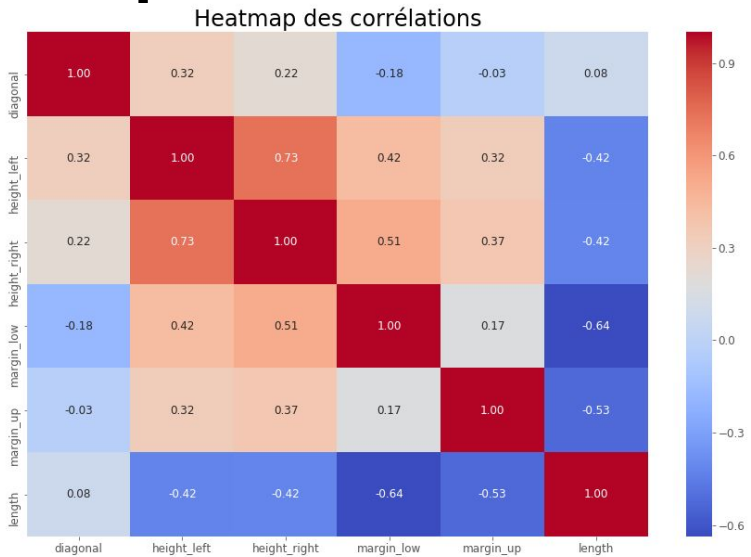
- 'margin' et 'height' ont des moyennes et médianes plus élevées pour les faux billets.
- 'length' a une moyenne et médiane plus basse pour les faux billets.
- 'diagonal' ne semble pas être significative.



Analyse des corrélations par Heatmap

Le Heatmap permet de visualiser les corrélations possibles entre nos variables.

Les ressemblances entre variables peuvent être approfondies par le calcul au cas par cas du coefficient de Pearson, ou par Scatterplot.




```
]: #Coefficient de corrélation linéaire de Pearson entre 'height_left' et 'height_right'
coef_height = st.pearsonr(data['height_right'], data['height_left'])[0]
coef_height
```

```
0.7343902682297874
```

La valeur obtenue est proche de 1, il existe donc une réelle corrélation linéaire entre ces deux variables. Concrètement, au plus 'height_left' aura une valeur élevée, au plus 'height_right' le sera aussi.

```
]: #Coefficient de corrélation linéaire de Pearson entre 'length' et 'margin_low'
coef_length_margin = st.pearsonr(data['length'], data['margin_low'])[0]
coef_length_margin
```

```
-0.6373516884716646
```

La valeur obtenue est proche de -1, il existe donc une corrélation linéaire négative entre ces deux variables.

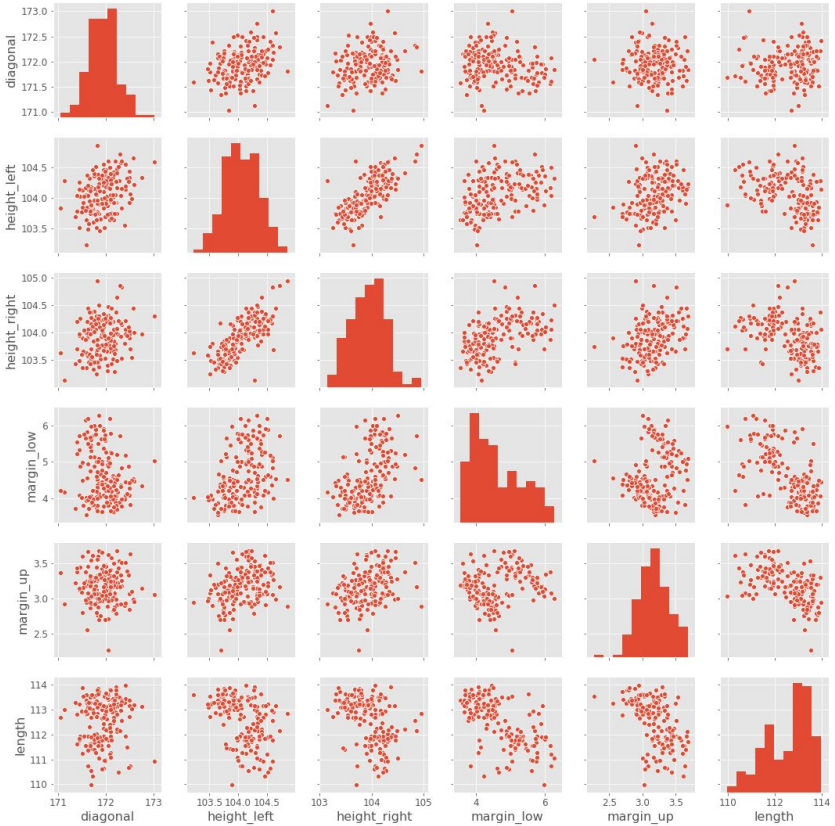
Visualisation des relations entre variables : diagrammes de dispersion des variables prises 2 à 2

La représentation par Pairplot permet d'avoir un aperçu rapide des liaisons entre variables.

L'exploitation graphique a été menée avec les précédents éléments de façon à avoir une vision des ressemblances et différences entre nos 6 variables :

- ➢ Corrélation positive des hauteurs droites / gauches
- ➢ Corrélation négative marge basse / longueur
- ➢ Corrélation négative hauteur / longueur
- ➢ etc…

Notre jeu de données comporte de nombreux cas de corrélation. Il existe une ou plusieurs redondances.



Traitement de la redondance des données

Notez que l'ACP est particulièrement utile lorsque les variables, dans le jeu de données, sont fortement corrélées. La corrélation indique qu'il existe une redondance dans les données.

En raison de cette redondance, l'ACP peut être utilisée pour réduire les variables d'origine en un nombre plus petit de nouvelles variables (= **composantes principales**), ces dernières expliquant la plus grande partie de la variance contenue dans les variables d'origine.

Analyse en Composantes Principales (ACP)

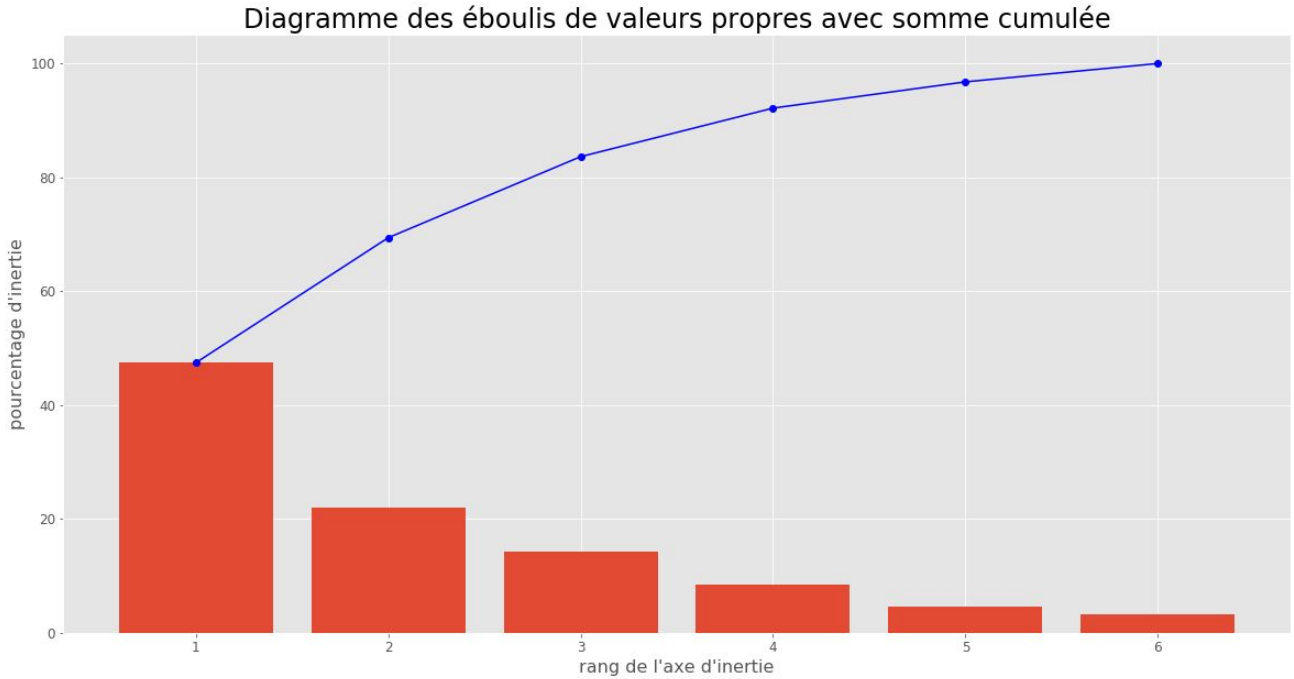
...

L'ACP (ou PCA en anglais) permet de dégager rapidement les principales tendances de l’échantillon, en diminuant le nombre de variables nécessaires à la représentation des données tout en perdant le moins d'information possible.

Combien de composantes à analyser ?

Ne pas perdre trop d'information, maximiser l'inertie, identifier par la «méthode du coude» les composantes nécessaires à l'atteinte des objectifs.

Ici les deux premières composantes principales seront retenues.



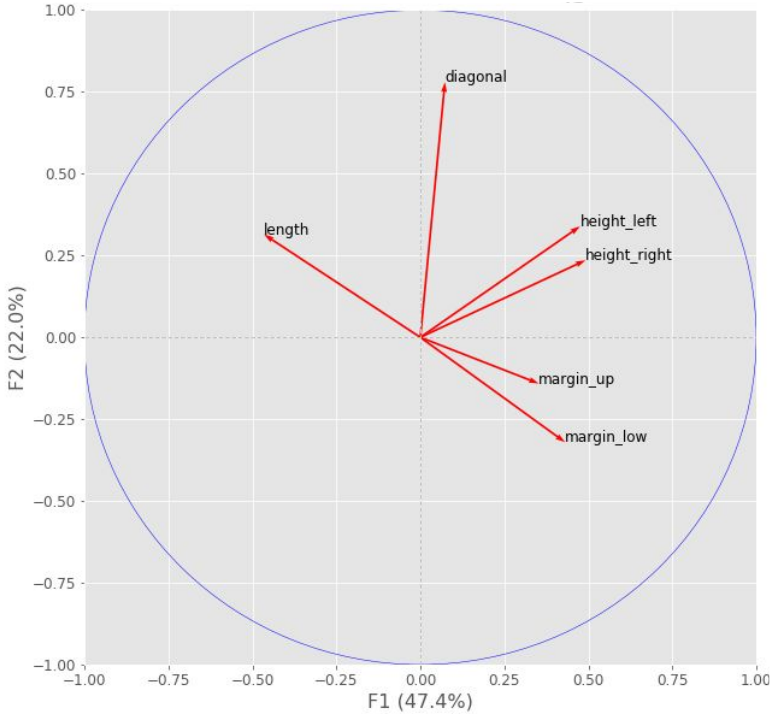
Cercle des corrélations : projection des *variables*

L'objectif est d'analyser les liaisons entre les variables. Identification des groupes de variables dans le premier plan factoriel.

Deux principaux groupes ressortent en F1, les hauteurs et les marges des billets.

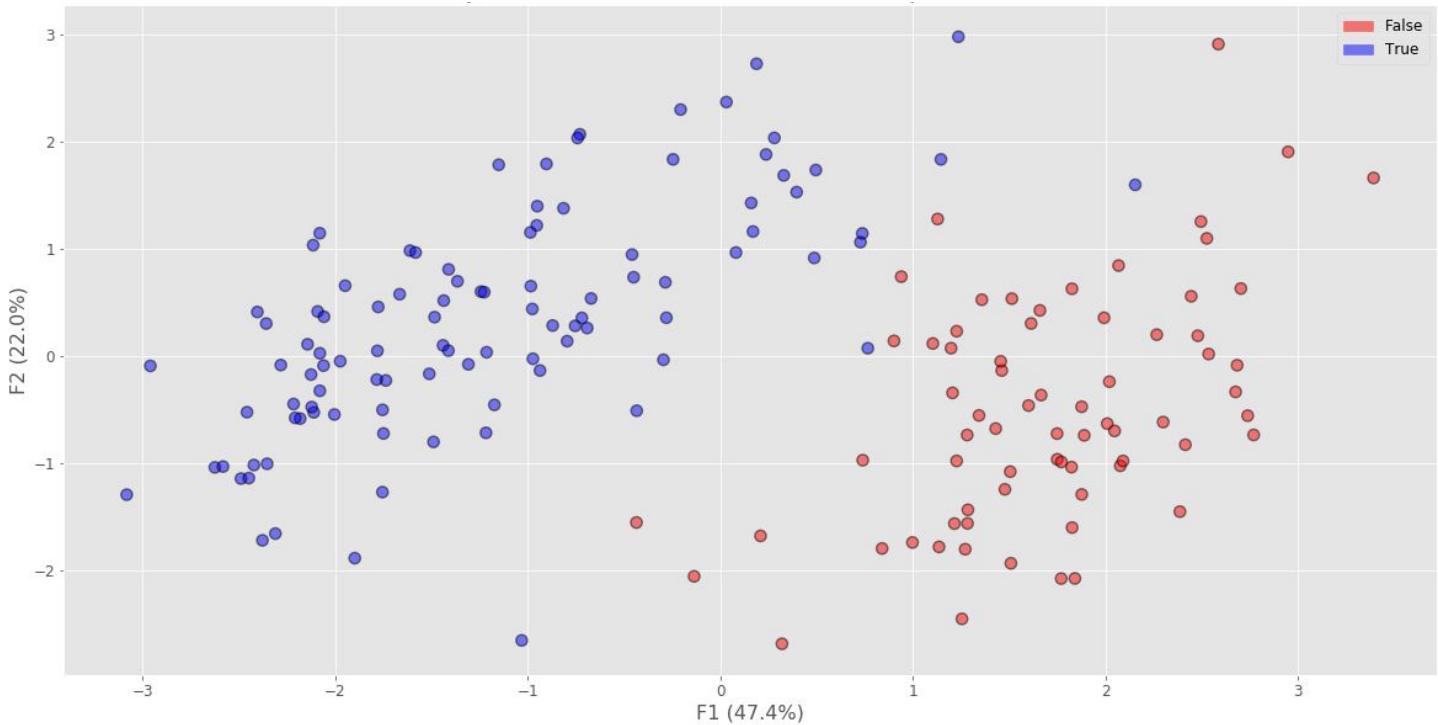
La longueur est corrélée négativement avec les deux précédents groupes.

La diagonale est mieux représentée en F2.



Projection des individus sur le 1er plan factoriel

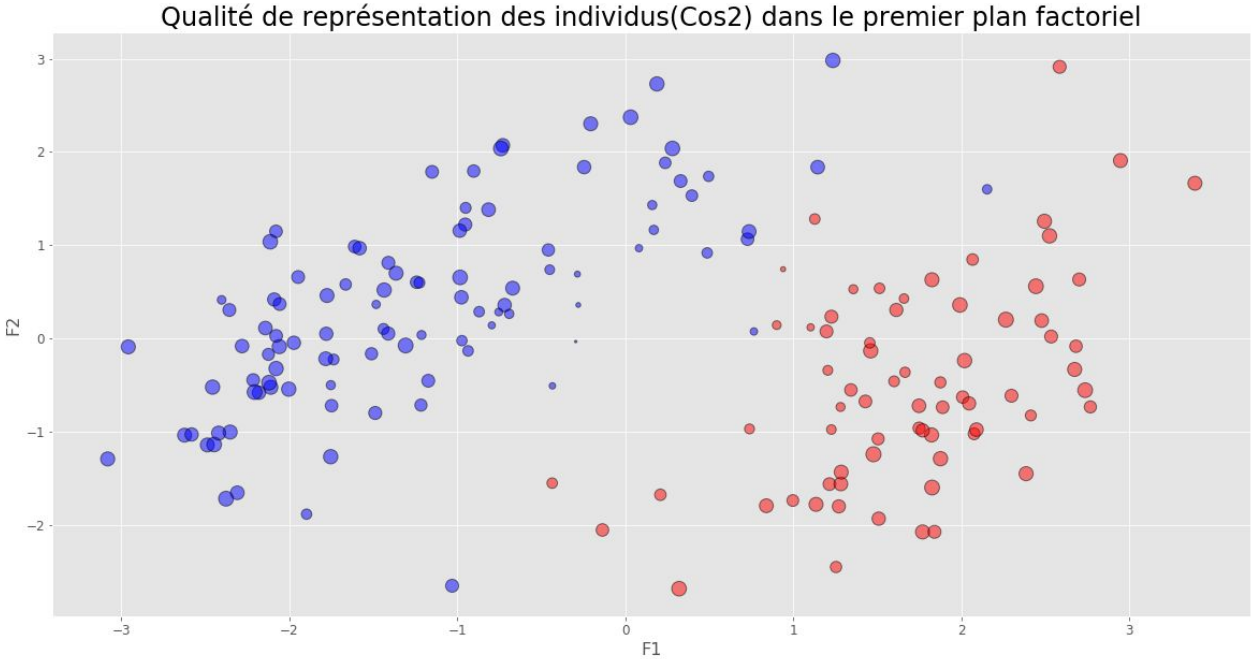
- L'axe F1 est celui qui sépare principalement les vrais billets des faux billets.
- L'axe F2 explique 22% de l'inertie, et participe à la représentation des billets.
- Une diagonale sépare clairement les vrais billets des faux billets.



Qualité de représentation des individus

Les individus les mieux représentés ont un cosinus proche de 1 (*angle proche de 0*). La représentation scatterplot sur le 1er plan factoriel, qui explique 69% de la variance, expose très clairement une grande majorité d'individus très bien représentés par les deux premières composantes.

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

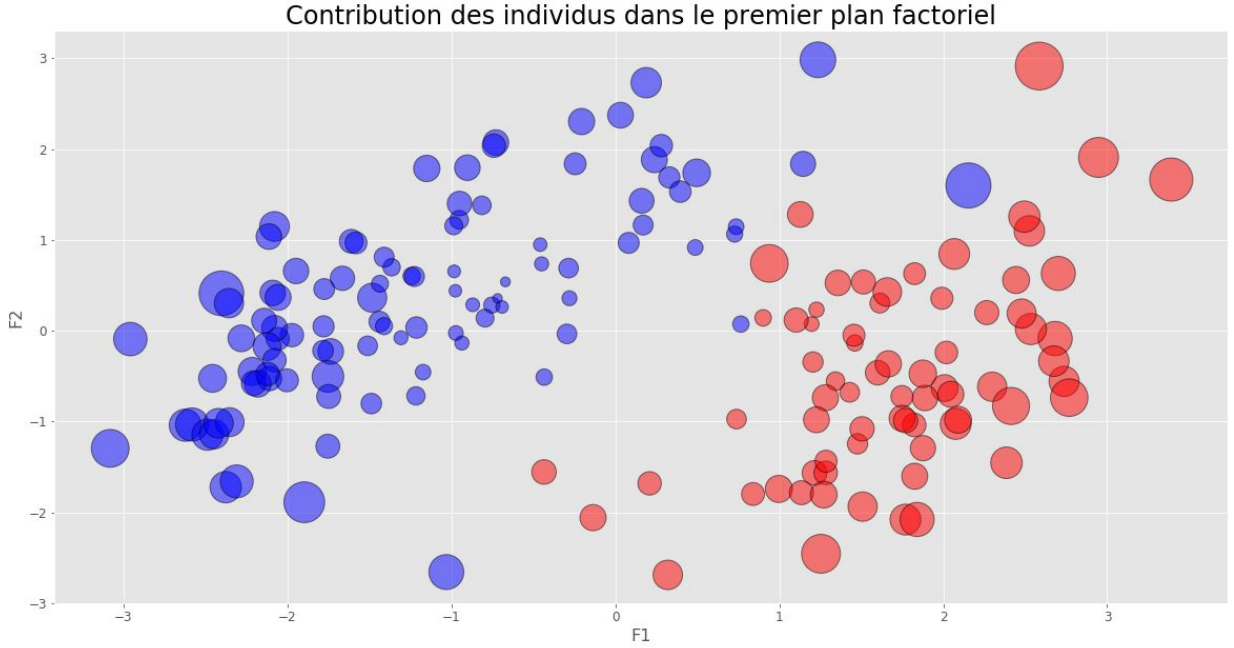


Contribution des individus dans l'inertie totale

Certains individus influent plus que d'autres sur le calcul des composantes :

- ceux qui sont le plus loin de l'origine à droite ou à gauche sont ceux qui contribuent le plus à F1.
- ceux qui sont les plus éloignés de l'origine en haut ou en bas sont ceux qui contribuent le plus à F2.

	c_inertie	is_genuine
0	18.410598	True
1	5.444799	True
2	4.960527	True
3	4.805719	True
4	18.039567	True
...	...	...
165	7.673233	False
166	20.625650	False
167	8.567410	False
168	3.482926	False
169	5.369633	False



Clustering K-Means

...

Le clustering apporte une classification des billets en fonction des caractéristiques quantitatives. Une fois le clustering réalisé, les clusters seront visualisés dans le premier plan factoriel de l'ACP.

Détection du nombre de classes k

Le **contexte métier** nous oriente vers un **k=2**, mais voyons si cette approche est optimale. Une stratégie simple pour identifier le nombre de classes consiste à faire varier K et surveiller l'évolution de l'inertie intra-classes. L'idée est de **visualiser le « coude »** où l'adjonction d'une classe ne correspond à rien dans la structuration des données.

Classification des billets par le K-Means

L'algorithme de clustering K-Means permet de **partitionner les données** en sous-groupes, par apprentissage non supervisée. Intuitivement, ces clusters regroupent entre eux des observations similaires.

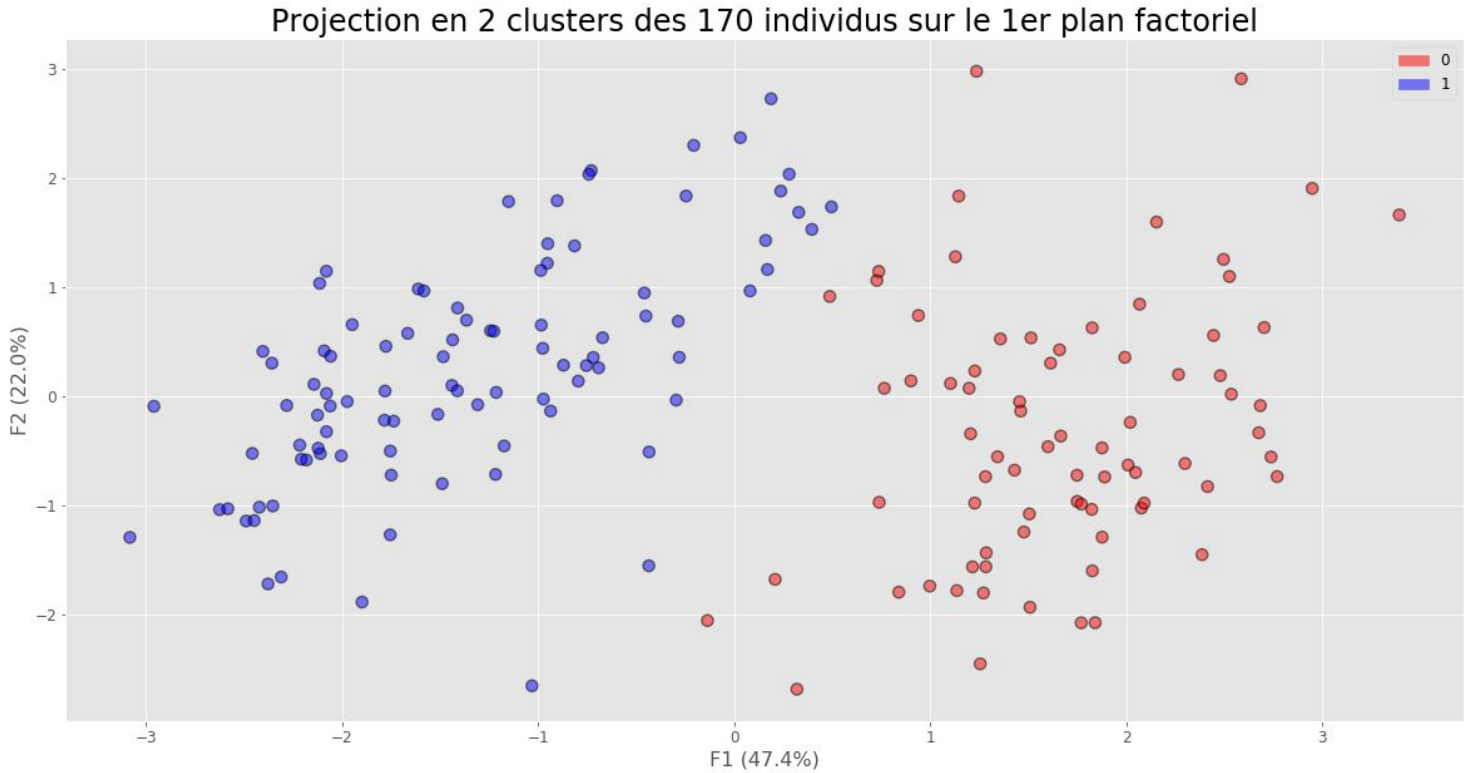
Dans scikit-learn, le k-means est implémenté dans *cluster.KMeans*. Par défaut, l'algorithme est répété 10 fois (paramètre *n_init*).

```
#Tableau des Centroïdes 2 clusters dans sa version centrée réduite
#La comparaison est tout de suite simplifiée, les dimensions prenant la même importance!
centroids = cls.cluster_centers_
pd.DataFrame(centroids, columns=df.columns)
```

	diagonal	height_left	height_right	margin_low	margin_up	length
0	-0.079185	0.645141	0.718611	0.840166	0.630841	-0.902645
1	0.064022	-0.521604	-0.581005	-0.679283	-0.510041	0.729798

Classification billets par le K-Means

La classification obtenue est très similaire au découpage effectué selon la matrice illustrative True/False. ***69% de la variance expliquée*** dans le 1er plan factoriel.



Centroïdes et correspondances clusters/nature billets

Les centroïdes permettent de mieux comprendre les différences et les ressemblances entre les deux clusters. L'apprentissage non supervisé K-Means permet d'obtenir des moyennes centrées réduites facilitant l'interprétation.

Le tableau des correspondances permet de mieux comprendre les résultats obtenus, ces classes sont quantifiées et traduisent la précédente représentation factorielle des 170 individus.

	diagonal	height_left	height_right	margin_low	margin_up	length
0	-0.079185	0.645141	0.718611	0.840166	0.630841	-0.902645
1	0.064022	-0.521604	-0.581005	-0.679283	-0.510041	0.729798

	Faux	Vrais
Cluster 0 - Faux	68	1
Cluster 1 - True	2	99

Modélisation par Régression Logistique

...

La régression logistique permettra de modéliser, de classifier la variable binaire «cluster» issue du clustering en fonction du corpus de variables quantitatives. Apprentissage supervisé de classification.

Méthodologie de régression logistique

On considère ici que l'on dispose de n observations d'un ***échantillon i.i.d*** «variables indépendantes et identiquement distribuées». Nous pourrons ainsi nous placer dans ***le modèle de Bernoulli***, et mettre en œuvre une ***Régression Logistique***.

```
]: #Préparation des features et des labels
   X = df_cls.copy()
   X = X.iloc[:, 2:]
   y = df_cls.iloc[:, 0]
```

```
]: #Séparation entre les données d'entrainement et les données test
   from sklearn.model_selection import train_test_split
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Ici, l'ensemble de données est divisé en deux parties dans un rapport de 20/80. Cela signifie que 80% des données seront utilisées pour la formation des modèles et 20% pour les tests des modèles. Les données étant assez limitées, le choix des 20% pour les données de test est choisi.

```
]: #Instanciation d'un modèle lr
   from sklearn.linear_model import LogisticRegression
   lr = LogisticRegression()
   lr.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Vérification de la colinéarité des variables

A partir du ***VIF (Variance Inflation Factor)***, le facteur d'influence de la variable, nous pouvons avoir une idée des éventuels ***problèmes de colinéarité***.

L'influence liée à la multicolinéarité des variables reste extrêmement faible, ici dans ce cas précis, ***le modèle de régression logistique n'a pas de problème potentiel de colinéarité.***

```
: from statsmodels.stats.outliers_influence import variance_inflation_factor

  reg_multi = smf.ols('diagonal ~ height_left + height_right + margin_low + margin_up + length', data=data).fit()
  variables = reg_multi.model.exog
  [variance_inflation_factor(variables, i) for i in np.arange(1,variables.shape[1])]
```

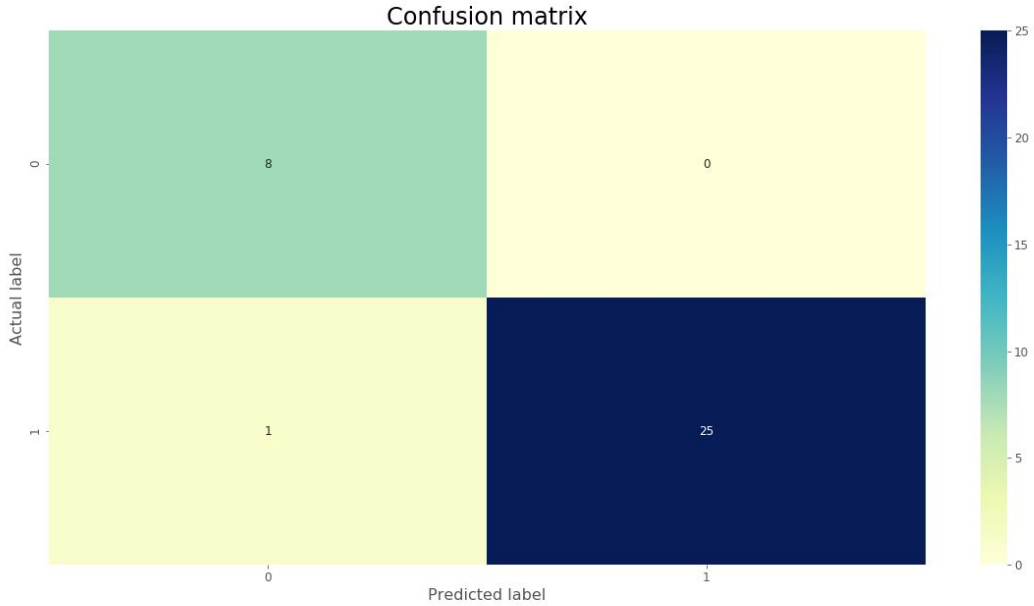
```
[2.249831212995896,
 2.5637087947433383,
 2.172802257913994,
 1.625230529655228,
 2.463836435483428]
```

Qualité du modèle par la Matrice de confusion

On peut évaluer son modèle par ***validation croisée***. Cela permet d'avoir une idée de la ***sensibilité*** et de la ***spécificité*** du modèle obtenu.

```
: from sklearn.metrics import confusion_matrix
  y_pred = lr.predict(X_test)
  confusion_matrix = confusion_matrix(y_test, y_pred)
  print(confusion_matrix)
```

```
[[ 8  0]
 [ 1 25]]
```



Courbe ROC et indicateur AUC

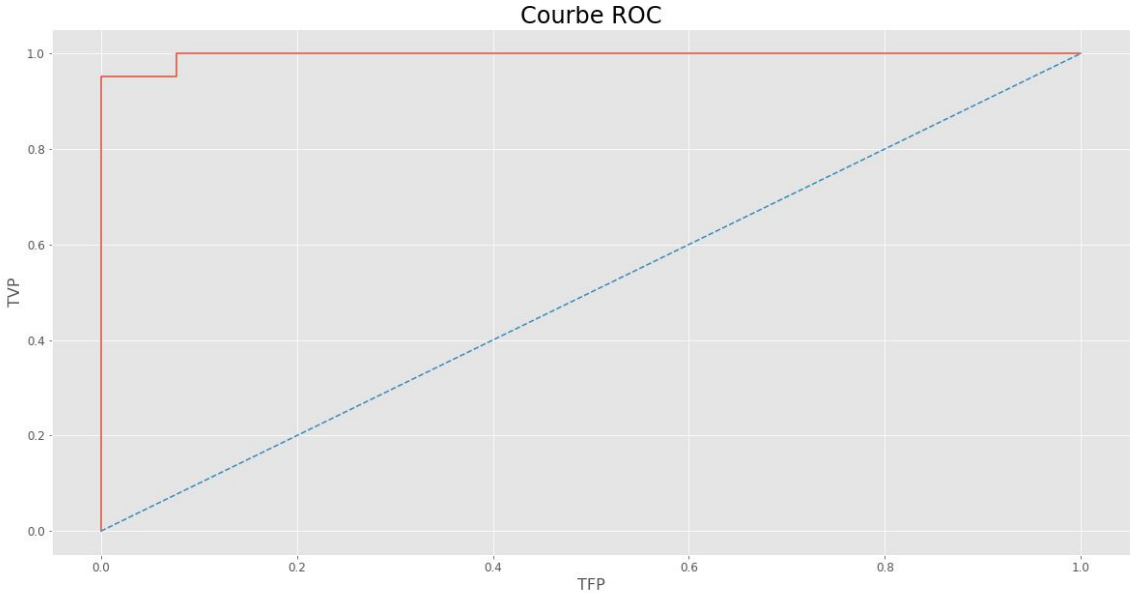
```
: # Récupération de la prédiction de la valeur positive
  y_prob = lr.predict_proba(X_test)[:,1]
  false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_prob)
```

```
: # Mesure AUC (Area Under the Curve): Aire sous la courbe
  from sklearn import metrics
  metrics.auc(false_positive_rate, true_positive_rate)
```

```
0.9963369963369964
```

Le principe est de faire varier le «seuil» de 1 à 0 et, pour chaque cas, calculer le taux de vrai positif et de faux positif.

Nous sommes très proche d'un schéma de *classifieur optimal.*



Autres indicateurs d'évaluation

```
#Autres métriques d'évaluation du modèle
print('  - Accuracy:' +str(round(metrics.accuracy_score(y_test, y_pred)*100, 2)), '%')
print('  - Precision:' +str(round(metrics.precision_score(y_test, y_pred)*100, 2)), '%')
print('  - Recall:' +str(round(metrics.recall_score(y_test, y_pred)*100, 2)), '%')
print('  - F1 score:' +str(round(metrics.f1_score(y_test, y_pred)*100, 2)), '%')
```

```
- Accuracy:97.06 %
- Precision:95.45 %
- Recall:100.0 %
- F1 score:97.67 %
```

Dans notre cas nous obtenons un ***taux de classification de 97.06%***, une excellente performance. Une ***précision très satisfaisante de 95.45%*** et une ***fonction Rappel de 100%*** indiquant la ***sensibilité de la régression***, c'est-à-dire que si des vrais billets font partie du groupe de test alors notre modèle peut l'identifier dans 100% des cas.

Application de l'algorithme sur un fichier test

L'algorithme de classification donnera la probabilité que le billet soit vrai.
Si cette probabilité est ***supérieure ou égale à 0.5***, le billet sera considéré comme ***vrai***. Dans le cas contraire, il sera considéré comme faux.

	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

```
#Préparation des données
X = df0.copy()
X = X.iloc[:, :-1]
```

```
#Utilisation du modèle de prédiction 'lr'
probability = lr.predict_proba(X.values)[:, 1]
```

```
#Probabilités des billets établies
proba = pd.Series(probability.round(3), name='value')
```

```
#Intégration des probabilités dans le jeu de données
df0_final = pd.concat([df0, proba], axis=1)
df0_final
```

	diagonal	height_left	height_right	margin_low	margin_up	length	id	value
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1	0.045
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2	0.016
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3	0.024
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4	0.844
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5	0.995

	diagonal	height_left	height_right	margin_low	margin_up	length	id	value	resultat
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1	0.045	Faux Billet
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2	0.016	Faux Billet
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3	0.024	Faux Billet
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4	0.844	Vrai Billet
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5	0.995	Vrai Billet

Lien vers l'algorithme de classification

...

A partir d'un fichier csv contenant 6 caractéristiques de billets nous pouvons désormais déterminer si chacun de ces billets est vrai ou faux, avec une probabilité associée : *lien vers l'algorithme**

**lien utilisé sur ma machine en local, sinon merci d'utiliser le fichier p6_02_algorithme_classification.ipynb*