Projet-6
Détectez des faux billets

Manu Sharma
Data Analyst (Openclassrooms)

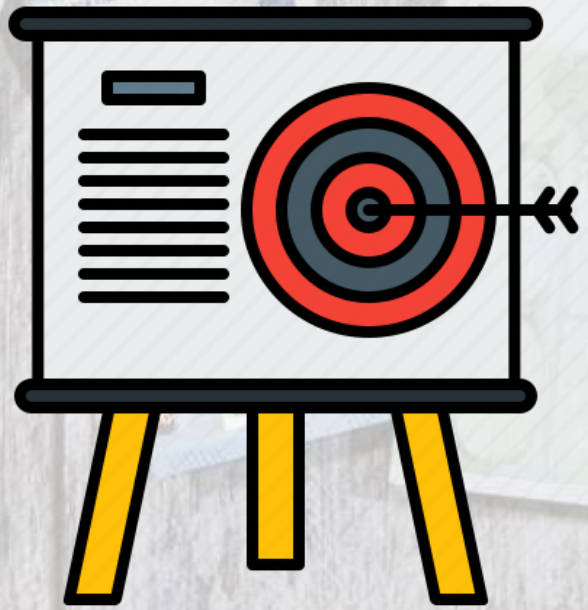
Objectif du Projet

Le scénario est:

Ma société de conseil en informatique me propose une nouvelle mission au ministère de l'Intérieur, dans le cadre de la lutte contre le crime organisé.

La mission qui m'est confiée:

Créer un ***algorithme*** de détection des faux billets qui permettra de traquer la fraude et de ***détecter les faux billets.***



Mission du Projet

Mission 0:

Description des ***données-analyses univariées et bivariées***

Mission 1:

Effectuer une analyse en composantes principales avec

analyse des valeurs propres ***(scree)***

représentation des variables par le ***cercle des corrélations***

représentation des individus par les ***plans factoriels***

analyse de la qualité de la représentation et de la contribution des individus**.**

Mission 2:

Algorithme de classification et analyse du résultat

Visualiser la partition obtenue dans ***le premier plan factoriel de l'ACP.***

Mission 3:

Prédiction du type de billet ***Modèle de Régression logistique***

Présentation du jeu de données

Données sur 170 billets

Les caractères des billets :

- ***Is_genuine:*** indique si un billet est vrai ou faux
- ***Length:*** longueur du billet en mm
- ***Height_left:*** hauteur mesurée à gauche du billet (en mm)
- ***Height_right:*** hauteur mesurée à droite du billet (en mm)
- ***Margin_low:*** La marge entre le bord inférieur du billet et l'image de celui-ci (en mm)
- ***Margin_up:*** La marge entre le bord supérieur du billet et l'image de celui-ci (en mm)
- ***Diagonal:*** la diagonal du billet (en mm)

Traitement et nettoyage des données

Ici, je prends les données du projet.

- ❖ Vérifier la longueur des données
- ❖ Vérifier les variables de données
- ❖ Vérifier les valeurs nulles
- ❖ Retirer les données catégoriques pour une analyse supplémentaire.

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("notes.csv")
```

```
df.head()
```

	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

```
df.tail()
```

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
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

```
list(df)
```

```
['is_genuine',
 'diagonal',
 'height_left',
 'height_right',
 'margin_low',
 'margin_up',
 'length']
```

```
df.isna().sum()
```

```
is_genuine      0
diagonal        0
height_left     0
height_right    0
margin_low      0
margin_up       0
length          0
dtype: int64
```

```
df.is_genuine.value_counts()
```

```
True     100
False     70
Name: is_genuine, dtype: int64
```

Data Cleaning

```
df_billets=df.drop('is_genuine',axis=1)
```

```
df_billets.head()
```

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

Mission 0
Analyses Univariées & Bivariées

Analyses Univariées

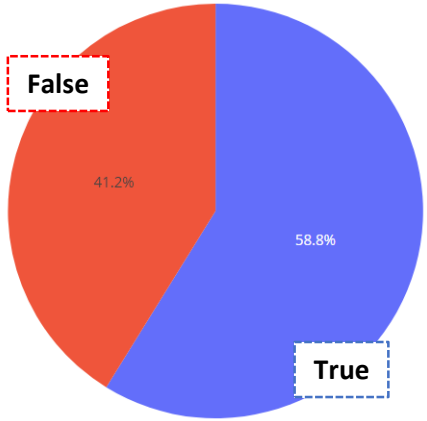
```
df_billets.describe()
```

	diagonal	height_left	height_right	margin_low	margin_up	length
count	170.000000	170.000000	170.000000	170.000000	170.000000	170.000000
mean	171.940588	104.066353	103.928118	4.612118	3.170412	112.570412
std	0.305768	0.298185	0.330980	0.702103	0.236361	0.924448
min	171.040000	103.230000	103.140000	3.540000	2.270000	109.970000
25%	171.730000	103.842500	103.690000	4.050000	3.012500	111.855000
50%	171.945000	104.055000	103.950000	4.450000	3.170000	112.845000
75%	172.137500	104.287500	104.170000	5.127500	3.330000	113.287500
max	173.010000	104.860000	104.950000	6.280000	3.680000	113.980000

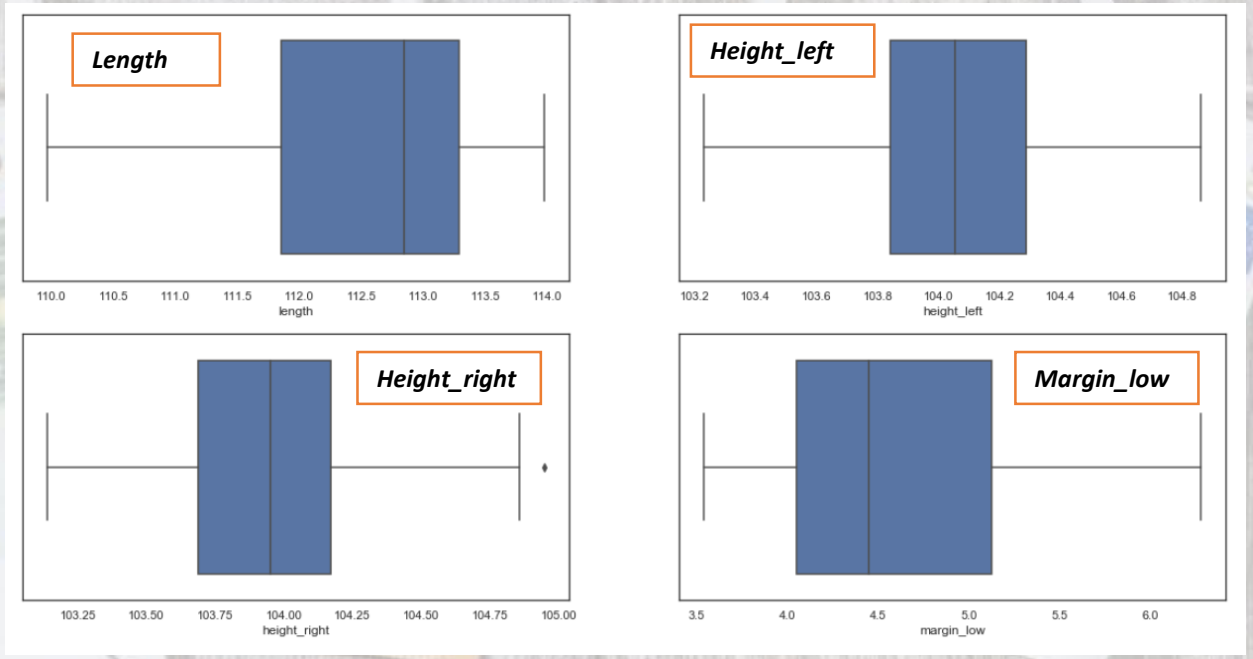
Description statistique des variables décrire ici la variable individuelle

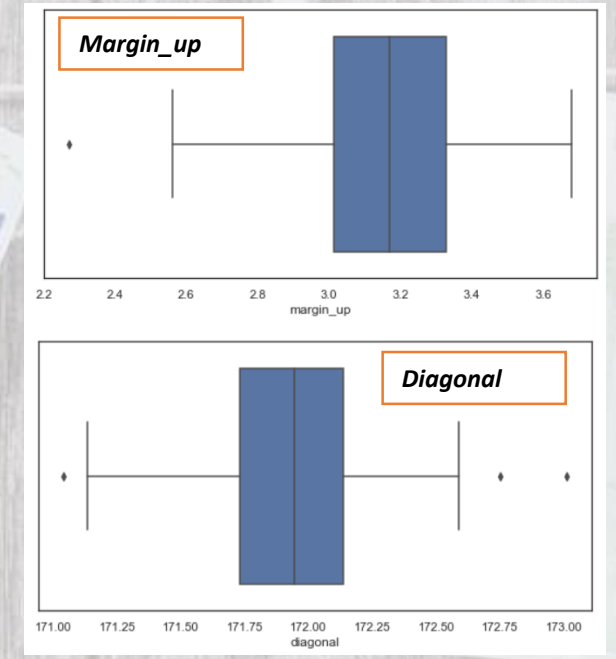
A l'aide de ce diagramme à barres, je peux trouver le pourcentage de la variable ***is_genuine***
% de billets authentiques : **58.2%**
% de billets faux : **41.2%**
Il n'y a donc pas de imbalance dans les données.

% of Billet Authenticity



Analyses Univariées avec Box Plot

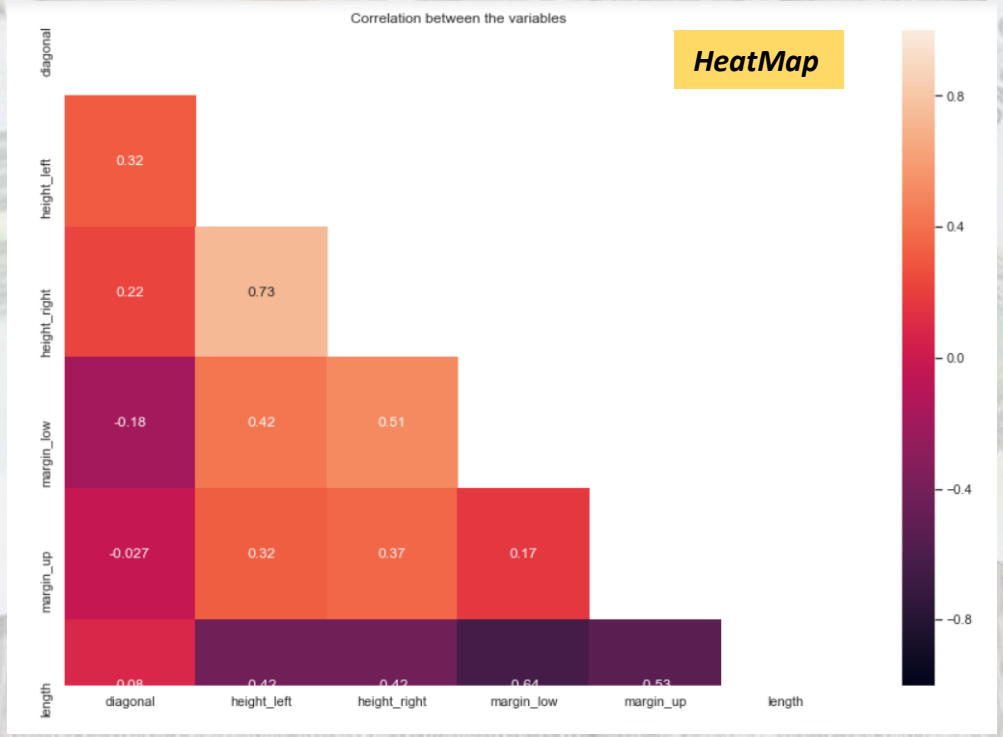




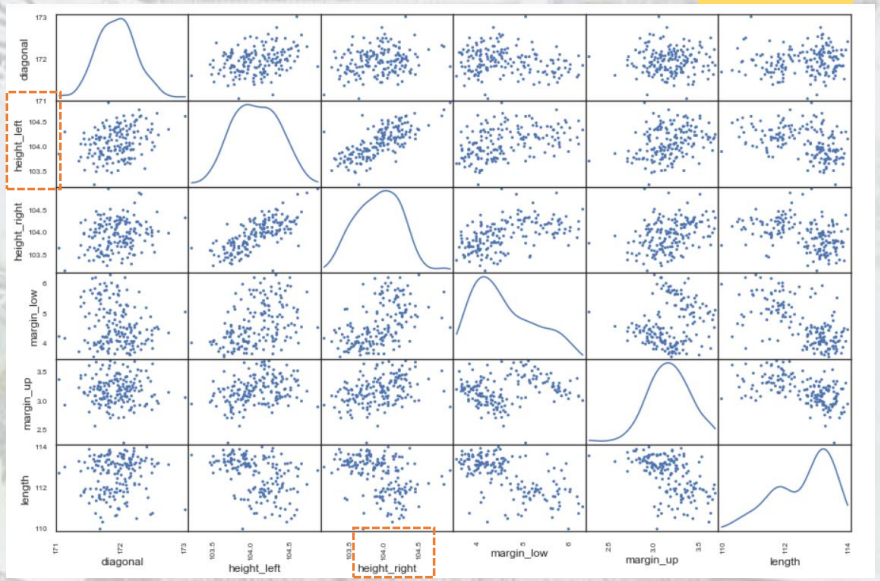
- Nous pouvons voir les quartiles et les valeurs aberrantes des variables à l'aide du diagramme en boîte
- Pour les variables ***Height left, right, diagonal et margin-up***, la distribution des données est symétrique (la médiane est au centre)
- Pour les variables ***Margin-low et Length***, la distribution des données est asymétrique

Analyses Bivariées

Corrélation entre les variables

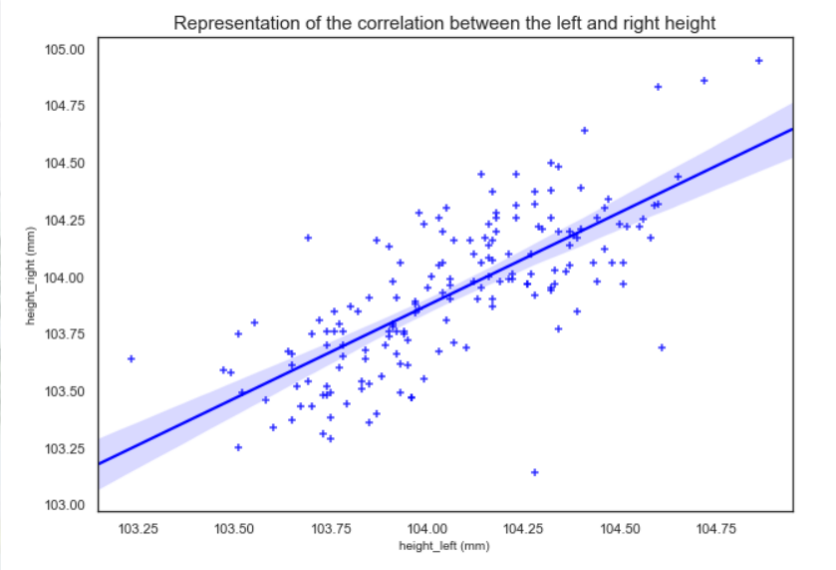


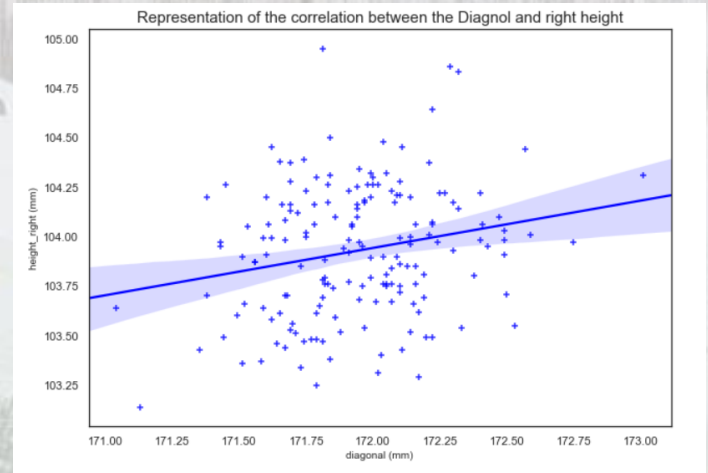
Pairplot

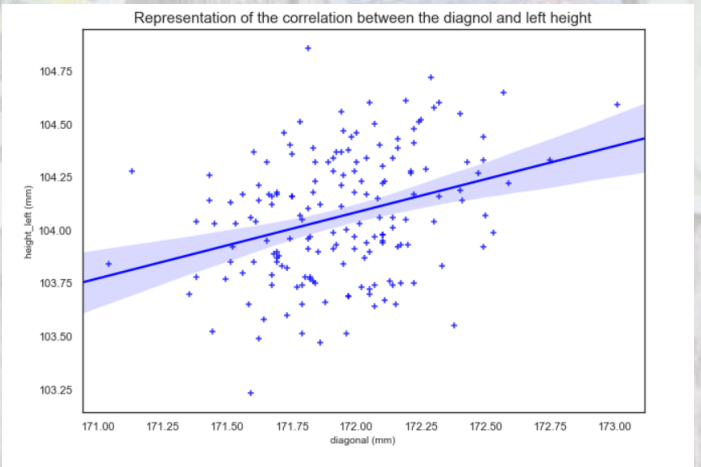


- Avec l'aide de la ***corrélation***, nous pouvons observer qu'il existe une ***corrélation positive*** entre la **height_left et la height_right**
- De plus, nous ne voyons aucune corrélation entre les autres varialbes comme : ***Margin_low, up avec Diagonal, Margin_up avec margin low***

Analyses Bivariées

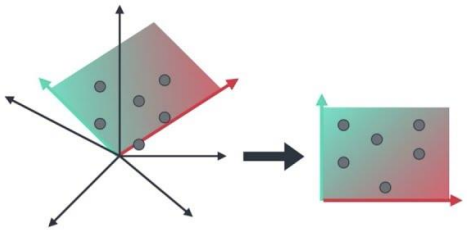






- Avec l'aide de la régression linéaire, nous pouvons voir la corrélation positive

- Ici, il y a une corrélation entre la diagonale et la hauteur, mais pas aussi forte qu'entre la hauteur à droite et à gauche



Principal Component Analysis

Mission 1

Analyse en composantes principales

Eigenvalue (Scree analysis)

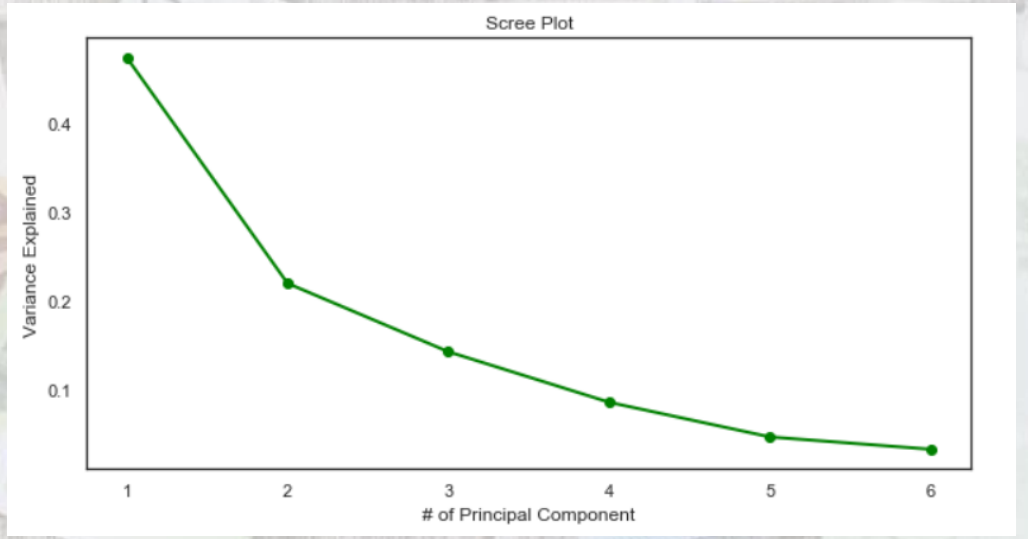
Nous utilisons ***l'ACP*** pour la réduction de la dimensionnalité. Ici, on essaie de préserver les variables essentielles qui ont plus de variation dans les données et de supprimer celles qui ont moins de variation.

Nombre de **PC** à retenir est une représentation graphique connue sous le nom de scree plot.

Il montre les valeurs propres sur l'axe des y et le nombre de facteurs sur l'axe des x.

La première composante explique généralement une ***grande partie de la variabilité***, les quelques composantes suivantes expliquent une quantité modérée, et les dernières composantes n'expliquent qu'une petite fraction de la variabilité globale.

Résultat : A partir du ***scree plot,*** nous pouvons analyser que les trois premières composantes principales contiennent la plus grande variance et stockent la plupart des informations.

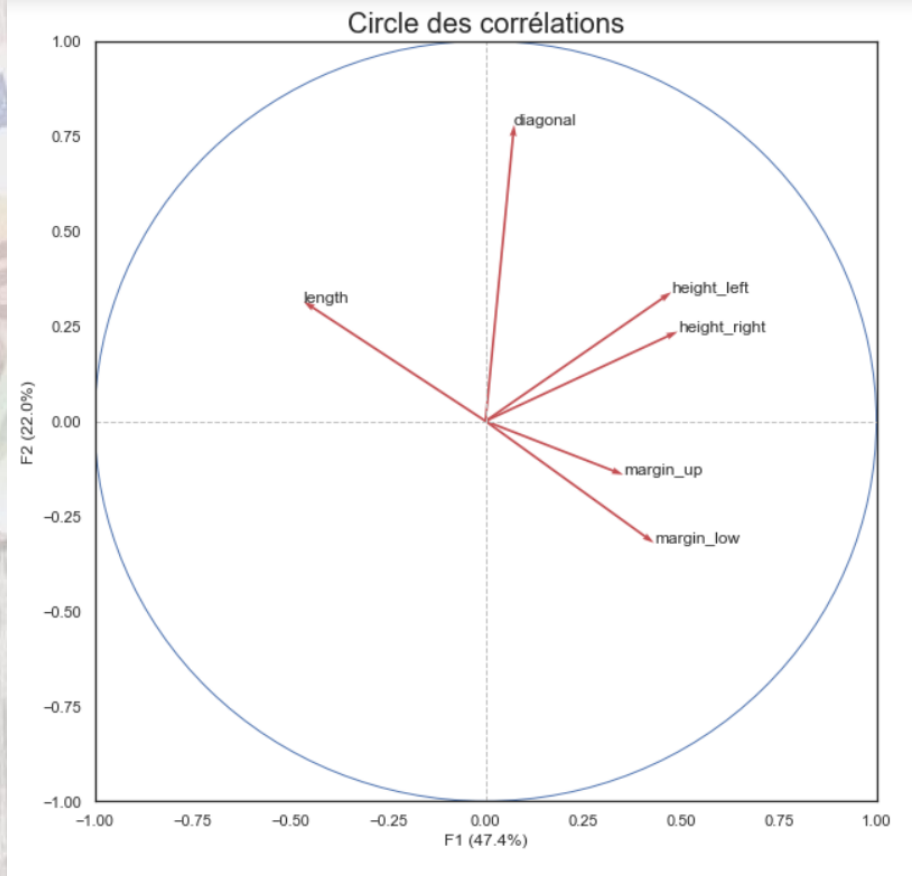


	Inertie %
comp 1	47.448
comp 2	21.957
comp 3	14.235
comp 4	8.526
comp 5	4.613
comp 6	3.221

Représentation des variables par le cercle des corrélations

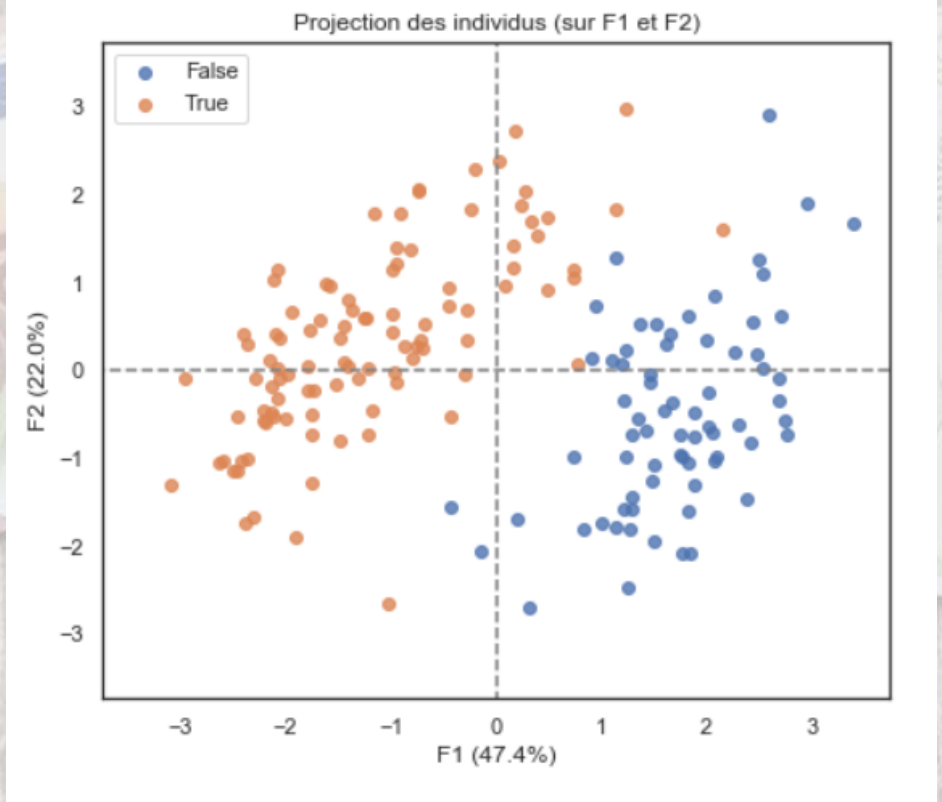
- ***Montrer les corrélations*** entre les variables et les corrélations entre variables et composantes principales.
- Si les flèches sont proches entre elles, les variables sont ***positivement corrélées.***
- Si les flèches ***négatives corrélées*** sont regroupées sur le quadrant opposé.
- Plus une flèche est proche du cercle plus elle est représentée par la composante principale.

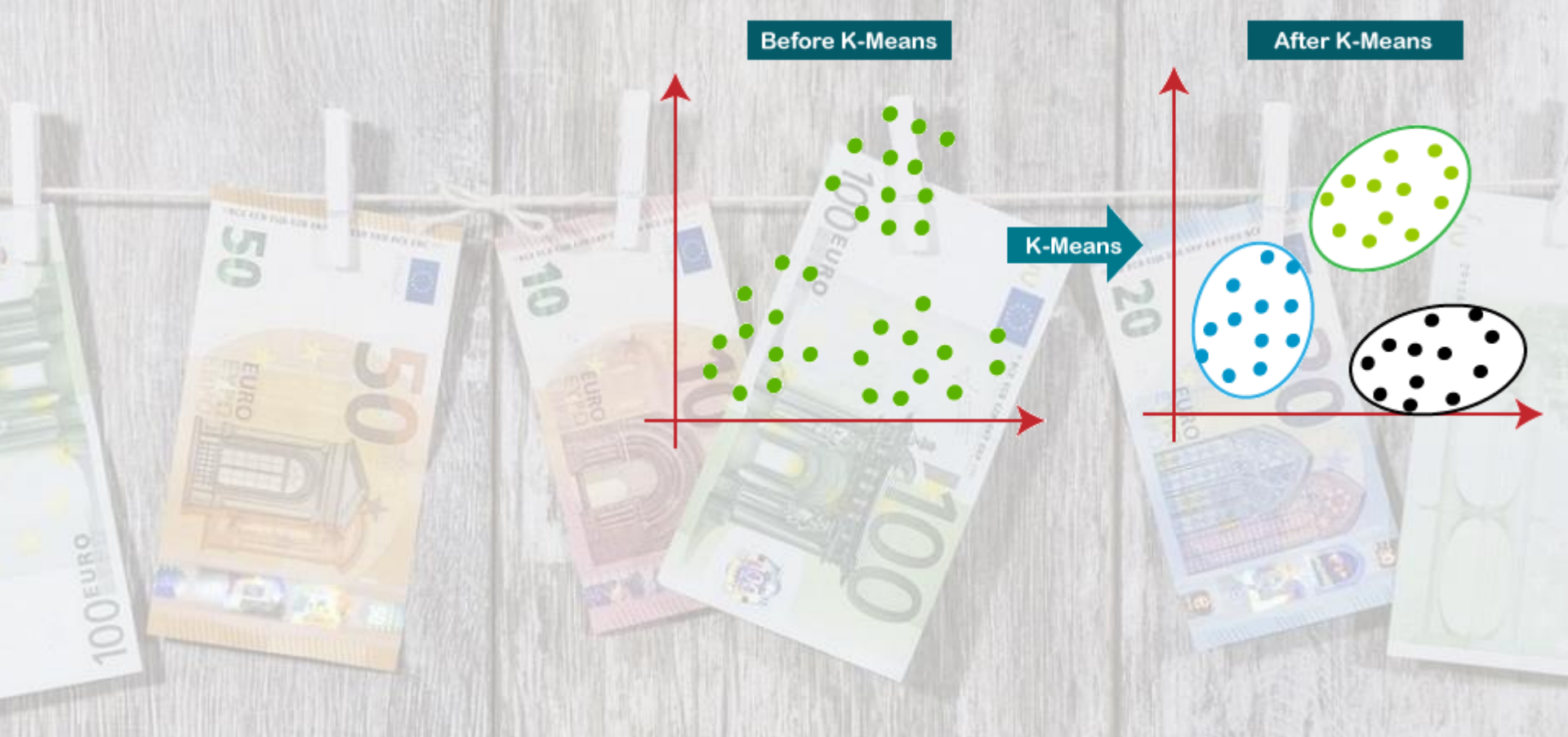
- ***Donc, c'est la conclusion de mon analyse précédente, il y a une corrélation positive entre height_left et right.***
- ***Et, aucune corrélation entre margin_up et low.***



Représentation des individus par les plans factoriels

- Comme nous pouvons le voir, les groupes séparés ***(Vrai et Faux)*** dans les plans factoriels, indiquant comment le ***premier plan factoriel*** a bien capturé l'essence des données.
- Le F1 donne **47.4% de variance**
- Et le F2 donne **22.2 % de variance**
- **70% de la variable par ces deux premiers plans factoriels.**





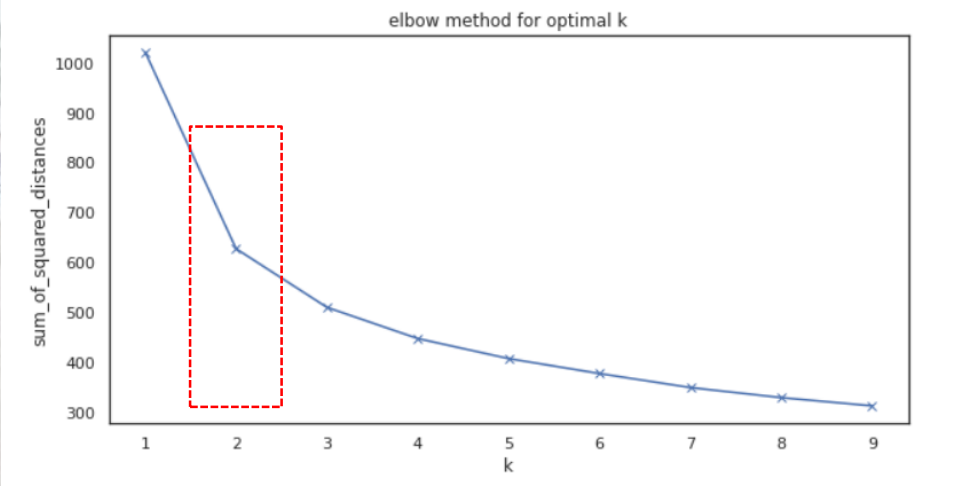
Mission 2
Algorithme de classification

Créer un modèle de clustering k-means

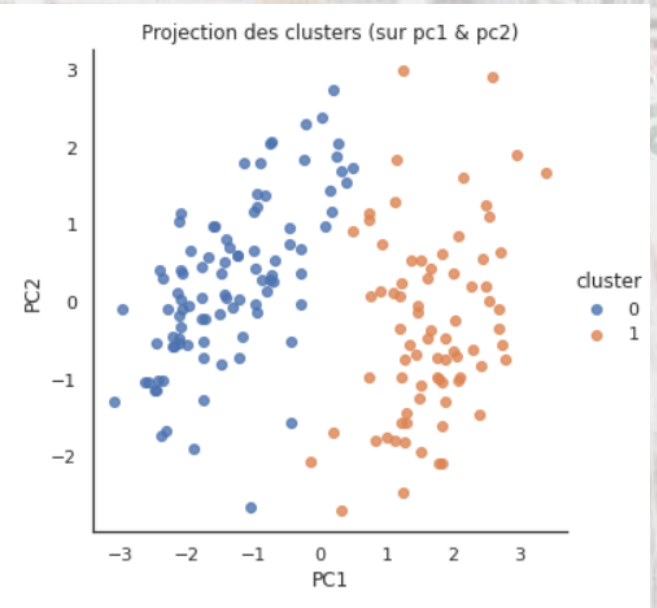
- Nous trouvons 2 clusters optimaux par ***clustering K-means (méthode du coude)***
- Analyser le cluster en le plaçant sur l'axe ***factoriel (composantes principales)***

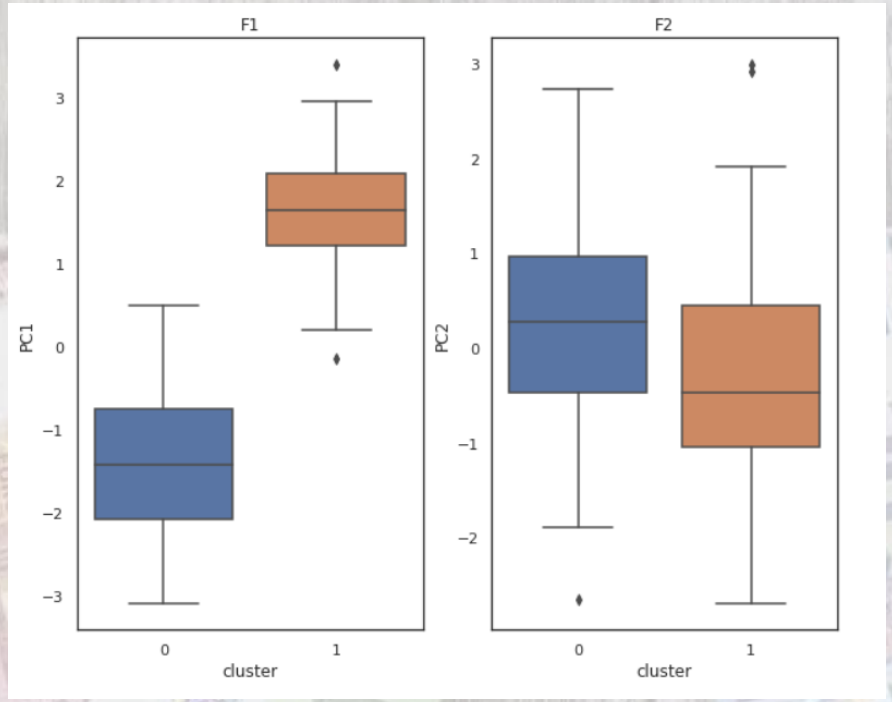
	diagonal	height_left	height_right	margin_low	margin_up	length	cluster
0	-0.428344	2.669454	3.096563	-0.131590	-1.189874	0.281633	1
1	-0.887559	-1.097697	-0.691254	-0.860126	-1.274740	0.780697	0
2	-0.362742	-1.030427	-0.509439	-0.303010	-1.232307	1.377405	0
3	-0.461145	-0.963156	-0.842767	-1.260106	-0.213913	1.149571	0
4	0.358883	-1.232238	-0.539742	0.611230	-3.820725	1.062777	0

	PC1	PC2	cluster
0	2.153639	1.599709	1
1	-2.110416	-0.526039	0
2	-1.973152	-0.048102	0
3	-2.059795	-0.089105	0
4	-2.403180	0.412170	0



Visualisez la partition obtenue dans le premier plan ***factoriel de l'ACP***



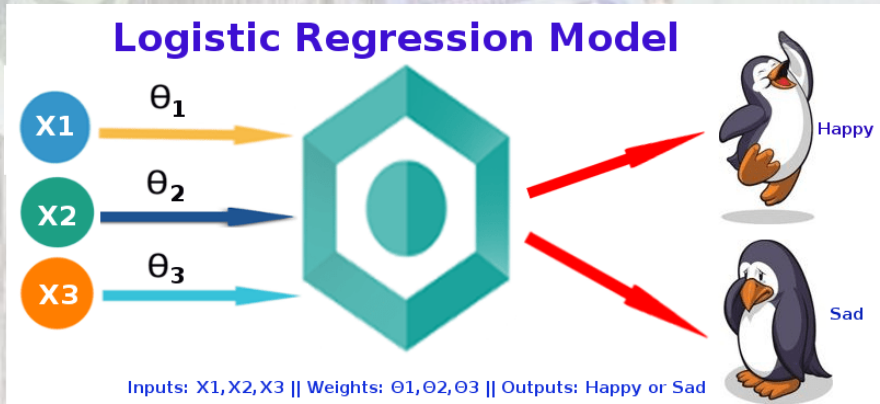


CONT..

	PC1	PC2	cluster
0	2.153639	1.599709	1
1	-2.110416	-0.526039	0
2	-1.973152	-0.048102	0
3	-2.059795	-0.089105	0
4	-2.403180	0.412170	0

- ***Sur Pc1,*** la grappe 1 a une tendance intéressante.
- ***Sur Pc2,*** la grappe 0 a une tendance attrayante

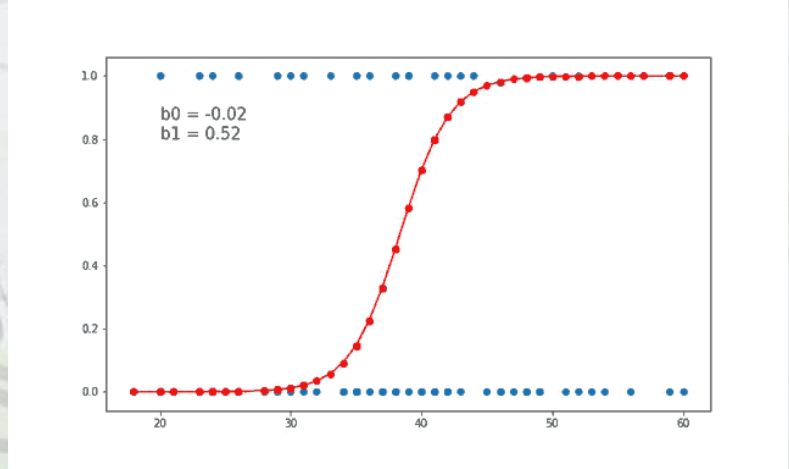
Mission 3
Régression Logistique.



Modélisez les données à l'aide d'une régression logistique.

Pour la régression logistique :

- D'abord je fais le modèle qui donne la probabilité qu'un événement (is_genuine) se produise en fonction des valeurs de la variable indépendante.
- Ensuite, j'estime la probabilité qu'un événement se produise
- Ensuite, prédire l'effet d'une série de variables
- Ensuite, classer l'observation en estimant la probabilité.
- **Ainsi, le modèle donne à la variable (is_genunine) une prédictibilité soit vraie soit fausse**



Régression logistique utilisation pour comprendre la relation entre la variable dépendante et une ou plusieurs variables indépendantes en estimant les probabilités à l'aide d'une équation de régression logistique. l'analyse peut vous aider à prédire la probabilité qu'un événement se produise ou qu'un choix soit fait.

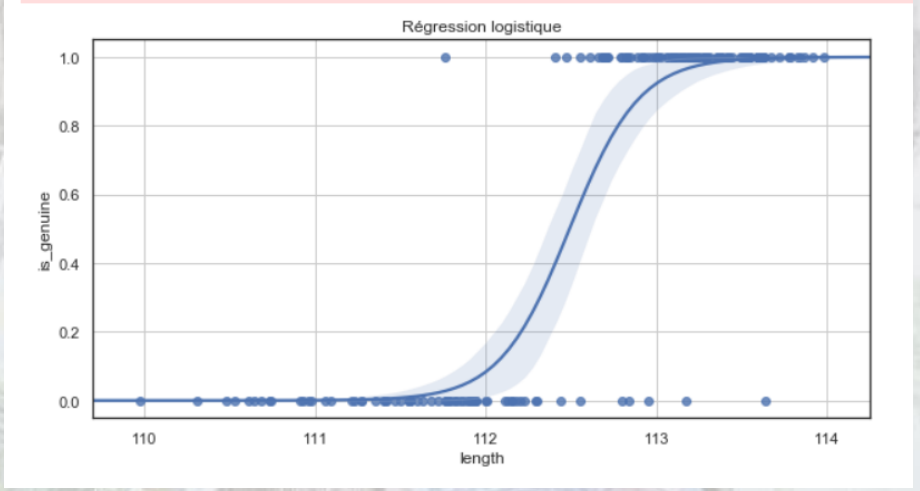
Pour préparer le modèle, je sélectionne la caractéristique

```
from sklearn.linear_model import LogisticRegression
X = data_model.drop(['is_genuine'], axis=1).values
y = data_model['is_genuine']
```

```
y.value_counts()
```

```
True     100
False     70
Name: is_genuine, dtype: int64
```

```
fig=plt.subplots(figsize=(10,5))
sns.regplot(x = 'length', y = 'is_genuine', data=data_model, logistic=True)
plt.title('Régression logistique')
plt.grid()
```



Divisé en deux ensembles de données, l'un pour l'train et l'autre pour le test

Je trouve que la précision de 100% du modèle

```
cat.score(x_test,y_test)
```

```
1.0
```

```
y_pred = cat.predict(x_test)
```

```
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(cat.score(x_test,y_test)))
```

```
Accuracy of logistic regression classifier on test set: 1.00
```

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42,stratify=y)
```

```
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
cat=LogisticRegression(solver = 'liblinear')
cat.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='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
cat.score(x_train,y_train)
```

```
0.9915966386554622
```

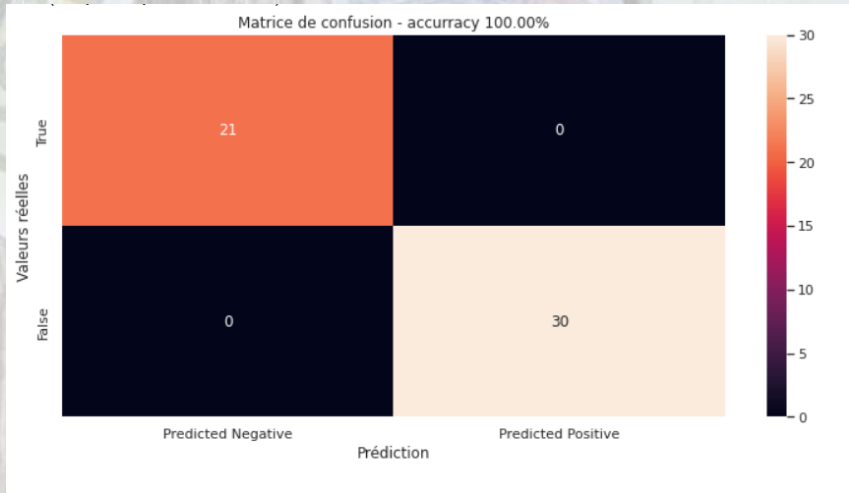
Évaluation du modèle par la matrice de confusion

	Predicted Negative	Predicted Positive
True	21	0
False	0	30

```
print(f"{reg_matrix.iloc[0,0]} et {reg_matrix.iloc[1,1]} est le nombre de prédictions correctes.")
print(f"{reg_matrix.iloc[1,0]} et {reg_matrix.iloc[0,1]} est le nombre de prédictions incorrectes.")
```

```
21 et 30 est le nombre de prédictions correctes.
0 et 0 est le nombre de prédictions incorrectes.
```

A partir de la matrice de confusion, nous avons la précision de 100% du modèle.
*Et, **trouver 21 et 30** est le nombre de prédictions correctes.*



Précision and Recall

- ***La précision*** est le rapport entre les vrais positifs et tous les positifs.
- ***Le recall*** est la mesure de notre modèle identifiant correctement les vrais positifs.
- ***Le recall*** donne également une mesure de la accuracy avec laquelle notre modèle est capable d'identifier les données pertinentes.
- ***La Accuracy*** est le rapport entre le nombre total de prédictions correctes et le nombre total de prédictions.

```python
# precision : tp / (tp + fp)
print("Precision:",metrics.precision_score(y_test, y_pred))

# recall : tp / (tp + fn)
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 1.0
Recall: 1.0
```

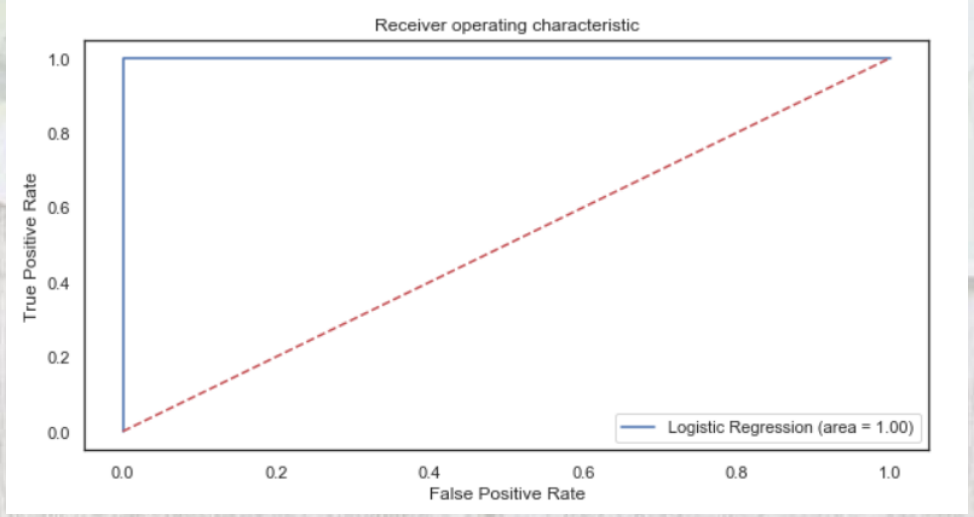
```python
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       1.00      1.00      1.00        21
        True       1.00      1.00      1.00        30

    accuracy                           1.00        51
   macro avg       1.00      1.00      1.00        51
weighted avg       1.00      1.00      1.00        51
```

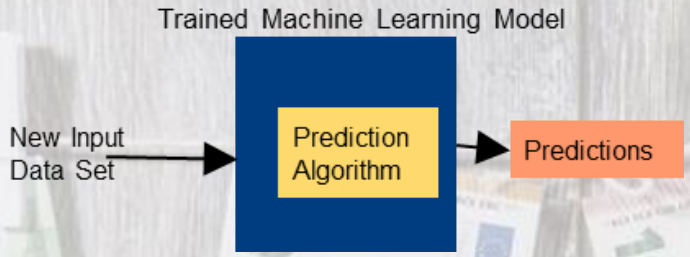
ROC (Receiver Operating Characteristic Curve)

- Comme la surface sous une courbe ***ROC*** est une mesure de l'utilité d'un test en général, où une surface plus ***grande signifie un test plus utile***, les surfaces sous les courbes ROC sont utilisées pour comparer l'utilité des tests.
- La zone délimitée par la courbe et les axes s'appelle l'aire sous la courbe (AUC).
- C'est cette zone qui est considérée comme une métrique d'un bon modèle. Cette métrique étant comprise entre 0 et 1, nous devons viser une valeur élevée de l'AUC.
- Les modèles avec une AUC élevée sont appelés modèles avec une bonne compétence.

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, cat.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test, cat.predict_proba(x_test)[:,1])
fig=plt.subplots(figsize=(10,5))
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



Model Prédictions




```
# Stockage de prédictions
predictions = cat.predict(x_test)
print(predictions)
#print(y_test)
```

```
[ True  True False False  True  True False  True  True  True  True False
 False  True False  True  True  True  True  True  True False  True
 False  True False False  True False  True  True  True  True  True
 False  True False False False False False False False  True False  True
  True  True False]
```

Voici la prédiction de mon modèle


```
print(y_test)
```

```
44      True
14      True
129    False
101    False
98      True
32      True
143    False
8       True
96      True
59      True
73      True
137    False
120    False
7       True
132    False
34      True
39      True
40      True
79      True
25      True
23      True
86      True
160    False
47      True
111    False
62      True
121    False
148    False
84      True
124    False
70      True
10      True
81      True
0       True
71      True
```

Merci